UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
FACULDADE DE CIÊNCIAS EXATAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

# **COLETA DIGITAL**: Desenvolvimento de protótipo de aplicativo móvel aplicado na coleta de dados de delineamento experimental

**Adolfo Fernandes Quaranta**

Diamantina - MG, Brasil
2017

UNIVERSIDADE FEDERAL DOS VALES DO JEQUITINHONHA E MUCURI
FACULDADE DE CIÊNCIAS EXATAS
DEPARTAMENTO DE COMPUTAÇÃO
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

# **COLETA DIGITAL**: Desenvolvimento de protótipo de aplicativo móvel aplicado na coleta de dados de delineamento experimental

**Adolfo Fernandes Quaranta**

Orientador:

**Prof. Dr. Alessandro Vivas Andrade**

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Diamantina - MG, Brasil
2017

**COLETA DIGITAL**: Desenvolvimento de protótipo de aplicativo móvel aplicado na coleta de dados de delineamento experimental

**Adolfo Fernandes Quaranta**

Orientador:

**Prof. Dr. Alessandro Vivas Andrade**

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

APROVADO em 29/08/2017

Prof. Dr. Alessandro Vivas Andrade
Orientador

Prof. Dr. Marcio Leles Romarco de Oliveira

Prof. Dr. Eric Bastos Gorgens

*Dedico este trabalho aos meus pais, Neusa e Ivan, irmãos, Isadora e Gabriel,*

*e à minha namorada Amanda, que juntos comigo sonharam,*

*e mesmo nos momentos mais difíceis,*

*sempre me apoiaram.*

# Agradecimentos

Agradeço primeiramente à Deus, por tornar tudo isso possível.

Ao meu professor Alessandro Vivas Andrade por toda dedicação, empenho e paciência ao me orientar durante este trabalho, e por todo conhecimento transmitido nas disciplinas lecionadas durante o decorrer do curso.

Aos professores, Marcio Leles e Eric Gorgens, do departamento de Engenharia Florestal desta Universidade, por compartilharem seus conhecimentos, tempo e toda ajuda necessária para a conclusão deste trabalho.

Agradeço também, aos meus professores, Alexandre Fonseca, Áthila Trindade, Caroline Queiroz, Cinthya Tameirão, Cristiano Pitangui, Eduardo Pelli, Euler Horta, Geruza Tomé Josiane Teixeira e Raphael Santin, pela imensurável contribuição para minha formação acadêmica com as aulas incríveis que sempre ministraram.

Aos amigos Ana, Camilla, “Dan”, “Deco”, “Ems”, “Grhég”, “Kel”, “Nanda”, “Naty”, Nicollas, “Nunes” e Tamara, que a Universidade fez as honras de nos apresentar e unir nessa caminhada.

À Pró-Reitoria de Assuntos Comunitários e Estudantis desta Universidade, onde estagiei e aprendi muito e compartilhando ótimos momentos com os amigos Adriano, Albér, Danielle, Elen, Kelly, Leida, Lidiane, Paulo, Renata, Márcia, Magno, Vanessa e Vânia.

Finalmente, agradeço à toda minha família. Aos meus avós maternos, Gentil e Creusa, e paternos, Estevam (*in memoriam*) e “Neném” (*in memoriam*), que sempre foram meus pilares e inspiração. Aos meus tios e tias, que sempre foram exemplos de luta e perseverança, e especialmente a aquelas que me receberam com tanto carinho quando me mudei para Diamantina, dinda Clara, tia Márcia e tia Nádia. E ao meu amigo irmão que sempre torceu muito por mim, Nielsen.

Muito obrigado à todos por fazerem parte deste sonho realizado.

# Resumo

Com o avanço e a popularização dos dispositivos móveis, se torna cada vez mais frequente o uso desta tecnologia para a solução de problemas nas mais diversas áreas do conhecimento. A agroindústria é um exemplo de campo que tem se beneficiado de forma muito direta com a aplicação de tecnologias móveis. Seguindo o contexto, este trabalho descreve as ferramentas e etapas seguidas na criação do protótipo de um aplicativo móvel. O app será utilizada para coleta de dados produzidos por delineamentos experimentais realizados para estudo e análise estatística. Além da coleta, o software faz o armazenamento e possibilita ainda editar os dados coletados ou exportar um arquivo compatível com diversos softwares estatísticos. Foi utilizado para construção do protótipo o Kit de Desenvolvimento de Software da plataforma Android, que exige a utilização da linguagem Java para programação das telas e lógica do software e do SQlite para construção e manipulação da base de dados. Ao final, é descrito um teste simples do software para apresentar as telas e as funcionalidades do protótipo.

**Palavras-chaves**: Android. Dispositivos Móveis. Coleta de Dados. Agroindústria.

# Abstract

Along to the advance and popularization of mobile devices, it's use become more and more frequent to solve problems in the most diverses fields of knowledge. The agroindustry is an example of area which has being directly benefited with application of mobile tecnology. Following the context, this article describes the tools and steps that where taken during the creation of a mobile application prototype. The app will be used to collect data of experimental desings performed to statistics study and analysis. Besides collection, the software stores and also allow edit collected data or export a file compatible to various statistics softwares. The Android's Software Development Kit was used to program the screens and software logic, which requires Java as programming language and SQLite to construction and manipulation of the data base. Finally, a simple test of the software was created to present the screens and functionalities of the prototype.

**Key-words**: Android. Mobile Devices. Data Collection. Agroindustry.

# Lista de ilustrações

# Lista de tabelas

# Lista de abreviaturas e siglas

| | |
|---|---|
| 1G | Primeira Geração |
| 2D | Bidimensional |
| 2G | Segunda Geração |
| 3D | Tridimensional |
| 3G | Terceira Geração |
| 4G | Quarta Geração |
| 5G | Quinta Geração |
| AMPS | Advanced Mobile Phone System |
| API | Application Programming Interface |
| APK | Android Package |
| App | Aplicativo |
| AVD | Android Virtual Device |
| BWA | Broadband Wireless Access |
| CDMA | Code Division Multiple Access |
| DBC | Delineamento em Blocos Casualizados |
| DIC | Delineamento Inteiramente Casualizado |
| DRM | Digital Restrictions Management |
| EV-DO | Evolution Data Optimized |
| EXP | Experimento |
| Gbps | Gigabit por segundo |
| GSM | Global System for Mobile |
| HAL | Hardware Abstraction Layer |
| HSPA | High Speed Packet Access |
| HTML | HyperText Markup Language |
| IMT | International Mobile Telecommunications |
| IP | Internet Protocol |

| | |
|---|---|
| IPC | Interprocess Communication |
| Kbps | Quilobit por segundo |
| km | Quilômetros |
| LTE | Long Term Evolution |
| Mbps | Megabit por segundo |
| MHz | Mega Hertz |
| MMS | Multimedia Message Service |
| PDA | Personal Digital Assistant |
| SDK | Software Development Kit |
| SGL | Scalable Graphics Library |
| SMS | Short Message Service |
| SO | Sistema Operacional |
| SSL | Secure Socket Layer |
| TDMA | Time Division Multiple Access |
| UMTS | Universal Mobile Telecommunication System |
| USB | Universal Serial Bus |
| VGA | Video Graphics Array |
| VM | Virtual Machine |
| WIFI | Wireless Fidelity |
| WiMAX | Worldwide Interoperability for Microwave Access |
| WLAN | Wireless Local Area Network |
| WVGA | Wide Video Graphics Array |
| XML | Extensible Markup Language |

# Sumário

# Introdução

Utilizar aplicativos moveis para a solução de problemas encontrados nas Ciências Agrárias vem se mostrando uma tendencia acessível e eficiente nos últimos anos. O desenvolvimento e utilização de aplicações para dispositivos móveis, em especial para Android, segundo Otsuka e Zanelato (2012) é exponencialmente crescente no cenário em que vivemos atualmente, pois necessitamos cada vez mais de soluções rápidas, funcionais, de baixo custo e que estejam sempre ao nosso alcance. “Dispositivos e tecnologias mobile são a tendência do momento, e terão grande impacto nas gerações futuras”, (SANTOS; SPIRLANDELLI; GOTARDO, 2012, p. 98).

Um exemplo do potencial da computação móvel no meio agrário é o trabalho de Bambini, Luchiari-Júnior e Romani (2014), que apresenta os resultados de uma pesquisa sobre aplicativos de Agrometeorologia publicados na loja de aplicativos do Android, a Google Play. Segundo o autor, multinacionais como a Basf e New Hollad desenvolveram aplicativos que estão entre os mais baixados e que tem como conteúdo uma mescla de informações sobre meteorologia e o mercado agrário, sendo assim, são viáveis para o uso dos produtores agrários brasileiros.

Outro ponto muito importante neste ambiente é a coleta de dados, que é utilizada em diversas áreas do conhecimento, e por isso desperta o interesse de pesquisas para criação e aprimoramento dos seus métodos de realização. Nos tempos passados, nossos ancestrais também realizavam estas coletas, registrando na forma disponível em suas épocas os dados extraídos de atividades por eles executadas. Pedras, papiro, madeira, metal, papel e outros materiais guardavam as informações colhidas da melhor forma possível, pois eram as mais atuais tecnologias criadas até aquele momento. No mundo atual, dispomos de tecnologia digital de ponta que pode ser utilizada para inovar e melhorar os métodos existentes de coleta, processamento e armazenamento de dados. De acordo com Waku et al. (2015), devemos dar importância para uma série de requisitos funcionais e não funcionais na coleta de dados, pois será de acordo com eles que a base de dados possuirá utilidade e credibilidade respectivamente.

A coleta de dados de qualquer ambiente em exploração, acontecimento cotidiano ou estudo é muito importante, pois é através dos dados obtidos que serão geradas informações

que por sua vez podem trazer diversos benefícios, como por exemplo, a melhoria na produção agrária e tecnológica. Devido ao estágio atual de avanço dos dispositivos móveis, das conexões de rede WI-FI, 3G/4G e de toda tecnologia, é possível obter e transmitir dados muito relevantes e precisos ou armazenar em bases de dados no dispositivo, conforme a necessidade. Podem ser capturados com estes aparelhos desde arquivos de multimídia, informações climáticas ou até mesmo o posicionamento geográfico, tudo em tempo real. A utilização de dispositivos móveis nas Ciências Agrárias não é uma técnica recente, como apresentado por Paula (2013), desde os PDAs, aplicativos móveis são desenvolvidos para este fim. Porém atualmente os dispositivos móveis, como citado anteriormente, são capazes de realizar aferições mais detalhadas e precisas, um exemplo disto é sistema desenvolvido pelo último autor citado acima, que é capaz de coletar dados georreferenciados para produtores agrários através de comandos de voz.

Ainda dentro do meio agrário, estão mais dois projetos, o primeiro foi desenvolvido por Junior e Ventura (2011), e tem por finalidade mapear a producão em uma determinada área, aferida por meio da utilização de um aplicativo móvel Android que é conectado via bluetooth com um aparelho de emissão de ondas sonoras que lê a produção da área determinada, com o levantamento de dados obtido, o sistema regula automaticamente a pulverização de defensivos para a área geográfica onde a produção foi baixa. O segundo projeto é um Sistema de Informações Integrado, desenvolvido por Pretto (2013), que utiliza um aplicativo móvel, e sistema web para controle da produção de uma granja.

Neste contexto, embasado pelas ideias das publicações citadas e de outros autores que serão abordados posteriormente no trabalho, aproveitando a tecnologia, a mobilidade e a conectividade providas pelos smartphones, em conjunto as funcionalidades do SDK do Android este trabalho irá descrever o processo de criação de um protótipo de aplicativo que automatiza a coleta e tabulação dos dados produzidos por delineamento experimental.

# 1 Discussão do Problema

## 1.1 Definição

A realização de coleta de dados estatísticos produzidos pelos experimentos desenvolvidos para análise estatística, em sua grande maioria, ainda é realizada pelo modo convencional, em papel, ou utilizado uma planilha eletrônica simples. Estes métodos de coleta utilizados atualmente estão sujeitos a muitos erros de preenchimento dos formulários. Por não existir uma interface que induz o usuário a inserir os dados de forma ordenada e intuitiva, a probabilidade de inserções equivocadas aumenta muito.

Quando utiliza-se o formulário de papel, outro problema que pode ocorrer é que durante a transcrição manual dos dados coletados em papel para o computador, aumentam as chances de que o digitador transcreva algum dado incorreto durante o processo.

Coletar os dados de experimentos de tamanho maior também pode demandar grande quantidade de tempo para ser concluído, neste caso, as planilhas também serão muito grandes, o que pode confundir a pessoa que está coletando os dados, causando erros de preenchimento.

## 1.2 Extensão e Profundidade

Para este trabalho, a extensão será especificamente ao âmbito da Universidade Federal dos Vales do Jequitinhonha e Mucuri, reduzida aos Cursos de Sistemas de Informação e Engenharia Florestal de Diamantina, onde do curso de Engenharia Florestal virá a demanda a ser solucionada, e do curso de Sistemas de Informação, a solução do problema por meio do uso de tecnologia e inovação.

O protótipo será desenvolvido para Sistema Operacional Móvel Android, pois é uma plataforma gratuita que disponibiliza muita informação para ajuda no desenvolvimento. O Aplicativo será basicamente composto por diversas telas com campos para entrada de dados, em sua maioria numéricos, mas existem também campos para texto. Os dados coletados serão

formatados e exportados para um arquivo em formato de tabela, como especificado em conjunto com o curso de Engenharia Florestal.

## 1.3 Histórico

Percebendo que existe uma solução viável, mais eficiente e fácil de ser utilizada para o problema de coleta de dados nos seus testes realizados em campo, a Engenharia Florestal propôs ao curso de Sistemas de Informação o trabalho de solucionar este problema com o uso da Tecnologia, especificamente com o desenvolvimento de um protótipo de aplicativo móvel para a substituição do modelo atual de coleta manual de dados.

## 1.4 Objetivos e Justificativa

Tem-se por objetivo substituir o método convencional de coleta dos dados por meio do uso da tecnologia, com o desenvolvimento de um protótipo de aplicativo Android, que tornará a coleta de dados mais rápida, precisa e segura. Além disso, com a informatização deste processo será também possível adicionar outras funcionalidades ao aplicativo, como por exemplo, reaproveitar um formulário de coleta já criado previamente no aplicativo para uma nova coleta ou até mesmo a eventual correção de dados de uma coleta já realizada.

Será também objetivo exportar os dados coletados em um formato de um arquivo de planilha, com formatação especial, que servirá de entrada para softwares estatísticos. Sendo assim, todo o processo de coleta, até a alimentação do softwares com os dados coletados, será mais eficiente e eficaz.

## 1.5 Metodologia

Será utilizado neste trabalho, todo o conhecimento adquirido durante o cursos de Sistemas de Informação, desde métodos ágeis de projetos de software, envolvendo levantamento de requisitos, prototipação, diagramação, criação de modelo relacional da base de dados, até as

melhores práticas de programação, utilizando frameworks para agilidade no desenvolvimento e todos os componentes necessários ao desenvolvimento protótipo de aplicativo.

# 2 Evolução da Computação Móvel

## 2.1 Dos celulares aos smartphones

Na Figura 1 está representada a evolução dos aparelhos celulares ao longo dos tempos até a chegada do primeiro smartphone, o iPhone da empresa Apple ®. Estes dispositivos agregaram cada vez mais funcionalidades e capacidade de processamento de informações, permitindo assim seu uso em diversas atividades e não somente para a telecomunicação como no início.


Figura 1 – Evolução dos Celulares (Sue Shellenbarger, 2012)

Simão (2011) relata que o telefone DynaTAC 8000X da Motorola foi o primeiro aparelho considerado celular. Lançado em 1973, possuía peso e tamanho assustadores para o padrão atual, pesava próximo de 1 quilo e media 30 centímetros, possuía apenas 1 hora de autonomia na bateria e capacidade para apenas 30 números telefônicos. Diferentes de hoje, os telefones celulares eram caros, pesados e não possuíam muitas funcionalidades.

Após este modelo, entre as décadas de 80, 90 e o início dos anos 2000 surgiram inúmeros outros, de diversas marcas como a própria Motorola, a Nokia, a IBM. Estes novos modelos chegaram com novas características como o peso menor e baterias com maior autonomia, possuíam também novas funcionalidades como calculadora, calendário, navegador web, mensagem instantâneas, entre outras.

Em 2007, surge um novo conceito para estes aparelhos, que com o passar dos anos foram

ganhando maior poder computacional, portabilidade e design passando a serem chamados de smartphone. A Apple ® foi quem apresentou esta revolução com um aparelho extremamente inovador para os padrões até então difundidos, a empresa batizou a sua criação de "iPhone". Uma das principais inovações deste aparelho foi o seu sistema operacional denominado iOS. O S.O. da Apple ® dava ao usuário do iPhone inúmeras possibilidades de personalização, manipulação de dados e o uso de aplicativos que agora eram desenvolvidos para o sistema operacional e não mais para um aparelho específico, como descreve Simão (2011).

Figura 2 – Steve Jobs apresenta o iPhone em 2007 (VEJA, 2017)

De acordo com Otsuka e Zanelato (2012), em resposta a Apple ®, em 2008, um consorcio formado por mais de 80 empresas do ramo de dispositivos móveis, fundado em 5 de novembro de 2007, chamada Open Handset Alliance, lançou o Sistema Operacional para dispositivos móveis Android. Esse sistema, tão funcional quanto o iOS, era mais barato, pois é baseado em um sistema aberto. Ainda em 2008, o "HTC T-Mobile G1" ou HTC Dream foi o primeiro smartphone a ser comercializado operando com o sistema Android.

Figura 3 – Apresentação do HTC Dream (G1, 2008)

## 2.2 Tecnologias de Redes Móveis

A grande popularização dos dispositivos móveis se deve muito ao fato do avanço alcançado pelas redes móveis. Inicialmente as conexões de rede eram somente cabeadas e então não se fazia possível o uso de rede nos dispositivos móveis. Contudo, surge o advento das conexões sem fio tornando mais fácil a comunicação e o acesso à informação nesses aparelhos. Na Figura 4 encontra-se a evolução das redes móveis ao longo dos anos em questão de padrões, velocidade de transferência de dados e da sua mobilidade.

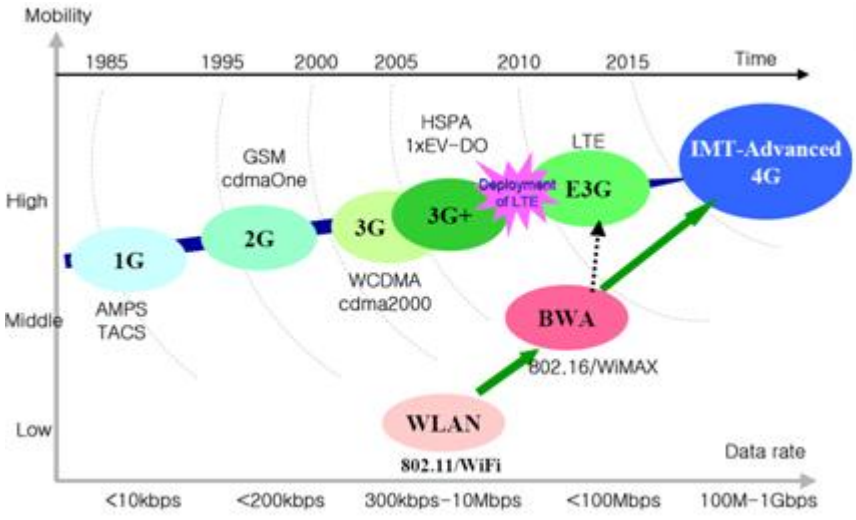



Figura 4 – Evolução das redes móveis digitais (TSENG et al., 2014)

### 2.2.1 Primeira Geração

Em meados dos anos 70 surge a primeira geração de redes celulares, uma enorme evolução se comparada às tecnologias de comunicação já existentes mas um pequeno passo comparada às próximas gerações. Com tecnologia analógica, as ondas de rádio eram de alcance muito curto e a transmissão de dados girava em torno dos 10Kbps. Por estas características, essa tecnologia não permitia nada além de chamadas de voz com pouca qualidade e segurança.

Segundo Jain et al. (2014), sua estrutura era baseada em dividir o terreno em células que variavam entre 10km e 25km e cada uma destas partes possuía uma estação base com antenas que retransmitiam em uma banda de ondas de baixa frequência. As células eram tão pequenas que a frequência podia ser reutilizada em células próximas, exceto em células adjacentes.


Figura 5 – Redes 1G (PAGOTO, 2016)

### 2.2.2 Segunda Geração

A segunda geração se apresenta para o mundo no final dos anos 80 com mudanças significativas, diferente da geração anterior, agora é utilizada tecnologia de múltiplo acesso digital o que superficialmente significa que o envio e recebimento dos dados pode ocorrer simultaneamente, como explica Descardeci (2001).

Com a utilização de tecnologias como TDMA (time division multiple access) e CDMA (code division multiple access), a segurança das transmissões foi melhorada e a taxa de transferência alcançou o patamar de 64kbps, trazendo a possibilidade do envio de mensagens de texto SMS (short message service) e ganho na qualidade das chamadas de voz.


Figura 6 – Redes 2G (PAGOTO, 2016)

### 2.2.3 Terceira Geração

Considerada como sendo uma família de padrões a terceira geração das redes móveis aparece em 2004, tornando melhor e mais vasto os serviços de transmissão de voz e de acesso a Internet via dispositivos móveis. Agora, a taxa de transmissão de dados podia alcançar até 2Mbps, o que possibilitou também a realização de chamadas de vídeo através da rede móvel.

Todo este avanço foi possível devido a modificações que países como Estados Unidos, Japão e o continente Europeu realizaram na tecnologia CDMA (Code Divison Multiple Access). Cada região nomeou seu padrão de forma diferente, mas como foram todas baseadas no CDMA são compatíveis entre si e foram considerados uma família de padrões. De acordo com Sharma (2013), na Europa o padrão foi chamado de UMTS (Universal Terrestrial Mobile System) enquanto nos Estados Unidos o nome escolhido foi CDMA2000.


Figura 7 – Redes 3G (PAGOTO, 2016)

### 2.2.4 Quarta Geração

Conhecida como 4G, a mais atual geração das redes móveis em funcionamento, teve seu primeiro teste de campo em Tokyo, no Japão, no meio do ano de 2005. Diferentemente das gerações anteriores, nesta o propósito foi a criação de um único novo padrão denominado LTE (Long Term Evolution). Com o LTE, visou-se entre outras questões a real padronização das redes de comunicação móvel 4G ao redor de todo o mundo, apesar disso, existem problemas de compatibilidade entre países/operadoras devido ao uso de diferentes frequências nas ondas de transmissão.

Ótimos resultados foram atingidos com a LTE, inicialmente pode-se destacar a possibilidade de utilização dessa tecnologia em veículos de alta mobilidade como trens e automóveis obtendo taxas de transferia de até 100Mbps. Para meios de baixa mobilidade como bicicletas, pedestres ou estacões de recepção fixas as taxas podem chegar em até 1Gbps. Para Jain et al. (2014), essas inovações não apenas melhoraram o acesso a Internet como também criaram possibilidade de novos comportamentos, como o uso de vídeo conferências e entretenimento em dispositivos móveis e também computação na nuvem, que é o acesso e compartilhamento de

arquivos pessoais em qualquer dispositivo conectado a Internet.


Figura 8 – Redes 4G (PAGOTO, 2016)

### 2.2.5 Quinta Geração

Essa que é a esperada próxima geração das redes móveis está em fase de desenvolvimento e é prevista para os próximos anos, mais especificamente 2020. Como dito por Sood e Garg (2014), a rede 5G vai abrir uma nova era na tecnologia de comunicação móvel. Os dispositivos 5G terão acesso à diferentes tecnologias sem fio ao mesmo tempo e o aparelho será capaz de combinar os diferentes fluxos das diferentes tecnologias. 5G é esperada para ser uma inteligente tecnologia capaz de interconectar todo o mundo sem limites.


Figura 9 – Redes 5G (PAGOTO, 2016)

# 3 Plataforma Android

A plataforma Android é composta pelo sistema operacional (Android SO), o SDK (Software Development Kit) e pelas aplicações(Apps). Utilizando-se o SDK, que é um conjunto de recursos e softwares de desenvolvimento, e a linguagem de programação Java, obtemos como produto um aplicativo Android. Neste capítulo todas estes conceitos serão explorados mais a fundo.

## 3.1 Sistema Operacional

Programado para controlar um computador e facilitar a interação do homem com a máquina, é o sistema operacional o responsável por executar um software e gerir todos os eventos que o usuário pretende realizar ao abrir um programa no computador. O SO realiza essa tarefa por meio de processos. “Um processo é basicamente um programa em execução”, (TANENBAUM; WOODHULL, 2008, p. 37).

O sistema operacional, como descrito por Oliveira, Carissimi e Toscani (2001), gerencia ainda a utilização de memoria pelos softwares em execução, controla os dispositivos de entrada e saída de dados, cria e organiza as informações no sistema de arquivos e também garante a segurança do sistema controlando a permissão de cada tipo de usuário .

Um sistema operacional móvel segue o mesmo conceito, porém está instalado em um computador com dimensões que permitem a sua portabilidade. Sendo assim, o Android é um sistema operacional móvel, pois foi projetado para prover todas as funcionalidades citadas acima e está instalado em um smartphone, que nada mais é do que um computador extremamente compacto e portátil, como diz Costa e Filho (2013).

### 3.1.1 Arquitetura

A arquitetura de um sistema operacional mostra como é a sua organização, como é mostrado para o Android na Figura 10. Neste sistema operacional móvel temos a arquitetura

de camadas, onde uma camada mais profunda na pilha provê suporte para as superiores. Cada camada presente no Android será melhor descrita abaixo, embasadas na sua documentação disponibilizada pelo Google Inc. (2016c).

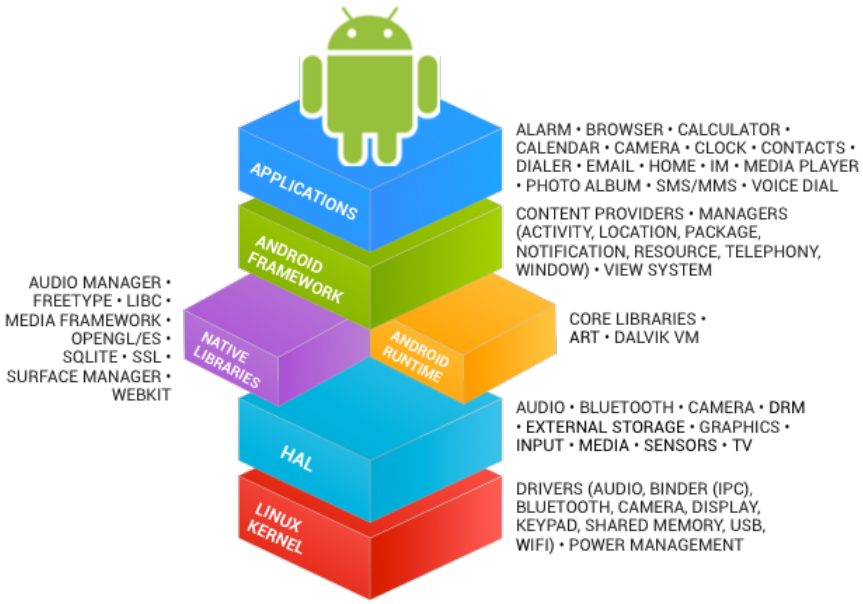



Figura 10 – Arquitetura da plataforma Android (Google Inc., 2016c)

#### 3.1.1.1 Camada Aplicações

Na camada Aplicações, é onde se localizam todos os aplicativos que são executados no sistema operacional, tais como, cliente de SMS e MMS, cliente de email, navegador, mapas, calculadora. Além apps incluídos no sistema, aqui também serão rodados aqueles que forem instalados por meio da Play Store e os desenvolvidos pelo usuário.

#### 3.1.1.2 Camada Android Framework

Esta camada é um framework de desenvolvimento padronizado e aberto que permite, com ajuda de provedores de conteúdo e outros serviços, a reutilização das funções das aplicações e seus recursos. Todas as APIs (Application Program Interface) disponíveis para o sistema principal também estão disponíveis para o desenvolvimento das aplicações, o que fornece ao desenvolvedor todos os recursos disponíveis do ambiente.

#### 3.1.1.3 Camada Bibliotecas Nativas

Na camada Bibliotecas o Android inclui um conjunto de bibliotecas C/C++ utilizadas por vários componentes do sistema. Essas bibliotecas também podem ser usadas nos aplicativos e são expostas para os desenvolvedores através do Framework. Na Tabela 1, estão algumas das principais bibliotecas:

| Biblioteca | Funcionalidades |
|---|---|
| LIBC | Uma implementação derivada da biblioteca C padrão, sistema (libc) do BSD, sintonizada para dispositivos rodando Linux. |
| Media Framework | Baseado no PacketVideo's OpenCORE, as bibliotecas suportam os mais populares formatos de áudio e vídeo, bem como imagens estáticas. |
| Surface Manager | Gerencia o acesso ao subsistema de exibição bem como as múltiplas camadas de aplicações 2D e 3D. |
| Webkit | Um web browser engine utilizado tanto no Android Browser quanto para exibições web. |
| OpenGL/ES | Para a versão 6.0 do Android a implementação é baseada no OpenGL ES 3.1 APIs; as bibliotecas utilizam aceleração 3D via hardware (quando disponível) ou o software de renderização 3D otimizado incluído no Android. |
| FreeType | Renderização de fontes bitmap e vector. |
| SQLite | Um poderoso e leve engine de banco de dados relacional disponível para todas as aplicações. |

Tabela 1 – Bibliotecas da camada Native Libraries

#### 3.1.1.4 Camada Android Runtime

Runtime é traduzido como tempo de execução em português, então temos nessa camada todas as bibliotecas necessárias para que o aplicativo Android seja executado da melhor maneira

possível. A execução dos aplicativos Android é feita sobre uma máquina virtual chamada Dalvik, e cada aplicativo possui o seu próprio processo e instância na máquina virtual. O Dalvik é otimizado para executar arquivos ".dex" (arquivo de classes compiladas em Java para a máquina virtual Dalvik), com o mínimo consumo de memória possível. Existe uma ferramenta chamada "dx" incluída no SDK Android que é responsável por ler os arquivos ".dex". O kernel do Linux não só é a base para o Dalvik como também realiza outras tarefas como gestão de baixo nível de memória e encadeamento.

#### 3.1.1.5 Camada de Abstração de Hardware (HAL)

Com a interface padrão e as bibliotecas disponíveis disponibilizadas nesta camada, é possível dar diferentes funcionalidades a um componente de hardware do dispositivo sem a necessidade de modificações nos drives ou no sistema, para isso é necessário criar abstrações(modelo em forma de software) que modificam programaticamente o comportamento deste hardware. Um exemplo de funcionamento disto pode ser com a interface de áudio, podem ser criadas duas abstrações, uma para a reprodução do áudio via Bluetooth e outra para reprodução via dispositivo de áudio USB, sem a necessidade de nenhuma outra modificação, o sistema será capaz de reproduzir corretamente o áudio.

#### 3.1.1.6 Camada Linux Kernel

A versão 8.0 do Android que é a última disponibilizada, utiliza a versão 3.18.48 do kernel do Linux para os serviços centrais do sistema, tais como segurança, gestão de memória e gestão de processos.

### 3.1.2 Versões Android

| Simbolo | Versão | Funcionalidades |
|---|---|---|
|  | Android 1.5 (Cupcake) | Conta com teclado virtual, widgets, upload de vídeos no YouTube e fotos no Picasa, som estéreo via Bluetooth. |

| Simbolo | Versão | Funcionalidades |
|---|---|---|
|  | Android 1.6 (Donut) | Melhora no desempenho, ferramenta de busca, suporte a resoluções WVGA, screenshot dos aplicativos na Google Market. |
|  | Android 2.0 (Eclair) | Suporte a HTML5, Bluetooth 2.1, otimização no desempenho, suporte a múltiplas contas, suporte ao Microsoft Exchange Server. |
|  | Android 2.2 (Froyo) | Ainda usada entre os dispositivos móveis disponíveis no mercado, conta com tethering e suporte a hotspot Wi-Fi, além de suporte ao Adobe Flash 10.1, atualização do Android Market. |
|  | Android 2.3 (Gingerbread) | Trouxe melhorias na interface, suporte a teclado multitoque, suporte a telas grandes e a resoluções maiores, acesso a múltiplas câmeras e gerenciamento de downloads. |
|  | Android 3.0 (Honeycomb) | Essa versão, que também foi otimizada para tablets, ganhou suporte a processadores multicore, a vídeochat e ao Google Gtalk, além de área de trabalho 3D. |
|  | Android 4.0 (Ice Cream Sandwich) | Nessa versão é unificada a plataforma de smartphones e tablets. O sistema conta com reconhecimento facial, suporte a mais dispositivos Bluetooth, novas funções de acessibilidade, mais opções de compartilhamento de arquivos em redes sociais. |
|  | Android 4.1 (Jelly Bean) | Conta com reconhecimento e digitação por voz offline, Google Maps offline, Google Now, nova tela de bloqueio, widgets mais eficientes, melhora nas mensagens de notificação. |
|  | Android 4.4 (KitKat) | As melhorias apresentadas nesta versão foram no desempenho multitarefa do sistema, tornando mais fácil o gerenciamento dos aplicativos abertos e exibição imersiva de conteúdos de mídia. |

| Simbolo | Versão | Funcionalidades |
|---|---|---|
|  | Android 5.0 (Lollipop) | A maior inovação apresentada nesta versão foi o Material Design, que trouxe uma nova aparência e novas melhorias relacionadas ao visual e usabilidade do Android. Outro ponto importante foi a alteração no tratamento das notificações, que agora podem ser personalizadas como desejar. |
|  | Android 6.0 (Marshmallow) | De acordo com o Google Inc. (2015a), no Marshmellow foram criados atalhos que fazem mais rápida a utilização do sistema, foram realizadas melhorias no gerenciamento de energia. Acompanhar e gerenciar o consumo de memória dos aplicativos também ficou mais fácil com o aprimoramento do sistema.<br>Além disso, a segurança do sistema foi melhorada com o gerenciamento de permissões, os aplicativos irão solicitar permissão de acesso aos dados ou aos sensores e câmeras do dispositivo apenas no momento em que delas necessitem. |
|  | Android 7.0 (Nougat) | Google Inc. (2016a) aponta que esta versão traz inúmeras melhorias, não só no quesito estético mas principalmente nas funcionalidades. Um dos destaques é a tela divida, onde será possível utilizar mais de um aplicativo ao mesmo tempo em uma única tela, outra melhoria seguindo este conceito é que o gerenciamento e a navegação entre os aplicativos abertos será melhorada ficando mais fácil de utilizar e rápida.<br>As notificações serão agora agrupadas de acordo com o seu aplicativo e o usuário poderá personalizar prioridade e sons para cada notificação. Para os aplicativos de mensagem, as respostas poderão ser escritas e enviadas diretamente da barra de notificação e tudo isso contando com novos emojis criados especialmente para essa nova versão do sistema. |

| Simbolo | Versão | Funcionalidades |
|---|---|---|
|  | Android 7.0 (Nougat) | Outras mudanças são que a instalação dos aplicativos será mais rápida, a economia de energia será maior devido a modificações no sistema de espera do Android e o download das atualizações do sistema serão em segundo plano, de forma que quando forem completamente baixadas, só serão instaladas na próxima reinicialização do sistema. |
|  | Android 8.0 (O - Preview) | A mais nova versão do Android, que ainda está disponível apenas para desenvolvedores, promete grandes modificações no tratamento de notificações. As principais novidades serão os canais de notificação, que irão permitir que o usuário crie categorias de notificações e ainda os badges de notificação, que mostram sobre o ícone dos aplicativos apenas as notificações especificas daquele app.<br>Ainda de acordo com o Google Inc. (2017a), diversas melhorias serão realizadas no sistema em áreas como conectividade, acessibilidade, compartilhamento, segurança e privacidade. |

Tabela 2 – Versões do Android

## 3.2 SDK

"O Android SDK fornece todas as ferramentas necessárias para o desenvolvimento de aplicativos Android" (DEITEL; DEITEL; WALD, 2016, p. 20). Atualmente existe um ambiente de desenvolvimento chamado Android Studio e integrado a ele estão ferramentas que em conjunto compõem o SDK. As ferramentas mais importantes como o gerenciador do SDK, o Emulador de dispositivos Android e todas as bibliotecas já inclusas para uso no desenvolvimento dos apps serão abordadas a seguir.

### 3.2.1 Android Studio

Inicialmente, quando o Android tinha acabado de surgir, seu ambiente de desenvolvimento padrão era o Eclipse, onde era adicionado um plugin para melhor integração com o

desenvolvimento Android. Contudo a Google vem aprimorando essa ferramenta ao longo dos anos e surge então o Android Studio. Desde 2015 esse ambiente de desenvolvimento integrado é oficialmente disponibilizado pela Google para o desenvolvimento de aplicações. Com essa ferramenta o desenvolvedor pode escrever o código fonte, debugar e testar a seu aplicativo de maneira intuitiva, rápida e segura.

Os desenvolvedores atualmente possuem com Android Studio uma diversa gama de possibilidades para facilitar o seu trabalho no desenvolvimento das aplicações. Na Figura 11 foram enumerados pontos especiais na interface com o usuário, que serão descritos na sequência.

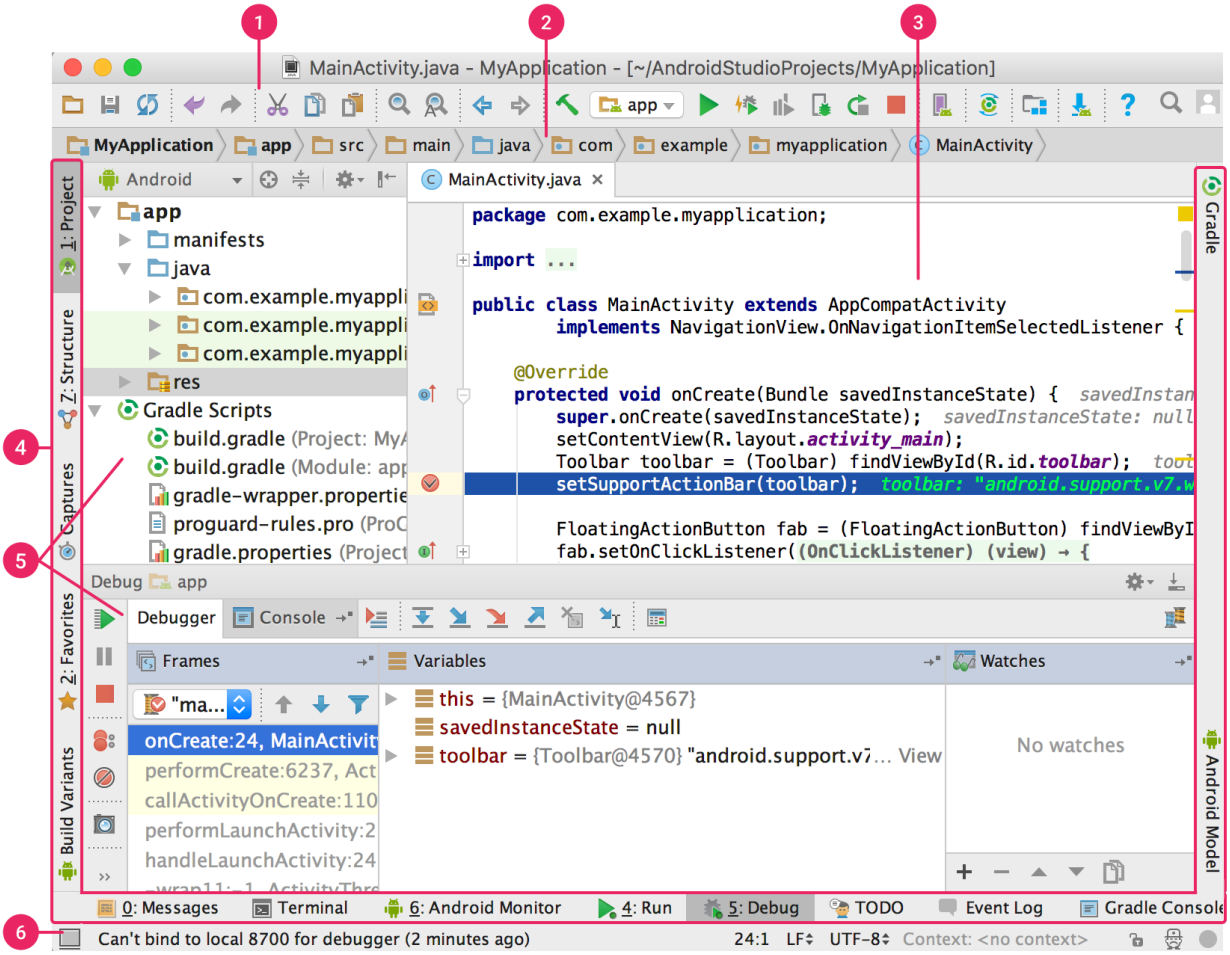



Figura 11 – Interface com o usuário no Android Studio (Google Inc., 2015b)

1. **Barra de ferramentas:** A partir da barra de ferramentas o desenvolvedor poderá acessar diversas ferramentas e funcionalidades do Android Studio. Daqui o desenvolvedor pode executar um projeto no dispositivo virtual Android por exemplo.

2. **Barra de navegação:** Nessa barra é exibido a estrutura de diretórios do projeto de forma compacta, por aqui e possível navegar entre alguns diretórios e arquivos do projeto.

3. **Janela do editor:** Aqui o desenvolvedor poderá visualizar ou editar o conteúdo escrito em qualquer arquivo de código que desejar. Essa seção irá variar sua apresentação de acordo com o tipo de arquivo em edição, por exemplo, quando se tem um arquivo de layout a exibição poderá ser da tela construída por aquele layout em questão.

4. **Barra de janela de ferramentas:** Para acesso à janela das ferramentas do Android Studio, basta que o desenvolvedor clique sobre o botão correspondente à uma das ferramentas exibidas nessa barra. Nela é possível acessar entre outras, as janelas de estrutura do projeto, de Debug, do Gradle ou do console.

5. **Janela das ferramentas:** As janelas das ferramentas podem ser expandidas ou recolhidas de acordo com a necessidade do desenvolvedor. Nessas janelas o desenvolvedor pode realizar tarefas como controle de versão do código, buscas, gerenciamento do projeto, debug do aplicativo entre outras funcionalidades.

6. **Barra de status:** Sua função é exibir mensagens sobre o status da IDE além de alertas ou avisos que merecem a atenção do desenvolvedor, como por exemplo, avisos sobre atualização da IDE ou do SDK.

#### 3.2.1.1 Estrutura do Projeto

Na aba de estrutura do projeto é onde se gerencia a estrutura da organização dos arquivos de código fonte, de recurso e de configuração do projeto do app. Por aqui é possível selecionar os arquivos que se deseja editar, criar novos diretórios ou arquivos necessários ao projeto.

Por padrão quando criado um projeto no Android Studio ele contém quatro divisões principais que como visto na Figura 12 são:

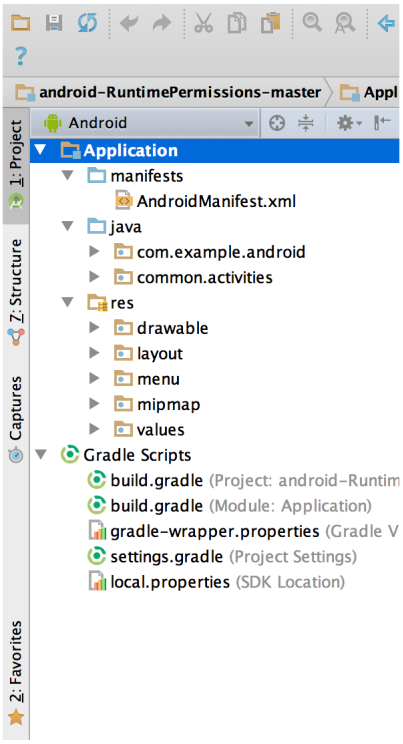



Figura 12 – Estrutura do projeto no Android Studio (Google Inc., 2015b)

- **manifests** - contém o arquivo de declarações das configurações referente ao aplicativo.
- **java** - contém os arquivos de código fonte e testes do aplicativo escritos em java
- **res** - contém os arquivos necessários que compõem o app mas que não são código fonte, por exemplo, imagens, layout das telas, strings, entre outros
- **Gradle Scripts** - contém os scripts de configuração do build do projeto (melhor detalhado em subseção 3.2.1.5).

#### 3.2.1.2 Gerenciador de Dispositivo Virtual Android

É possível desenvolver aplicações Android mesmo não possuindo um dispositivo móvel físico rodando o SO da Google. No Android Studio é disponibilizada uma ferramenta chamada AVD Manager, que traduzido do inglês significa Gerenciador de Dispositivo Virtual Android.

Com o AVD Manager, todas as versões do Android podem ser emuladas através de um sistema que cria virtualmente um dispositivo móvel, rodando a versão Android de sua preferência e com todas suas funcionalidades presentes como em um dispositivo físico. Assim, o desenvolvedor pode criar, rodar e testar as aplicações de forma completamente fácil e gratuita.

Para iniciar o AVD Manager, basta clicar sobre o ícone em formato de smartphone que pode ser encontrado na barra de ferramentas do Android Studio, veja na Figura 11.

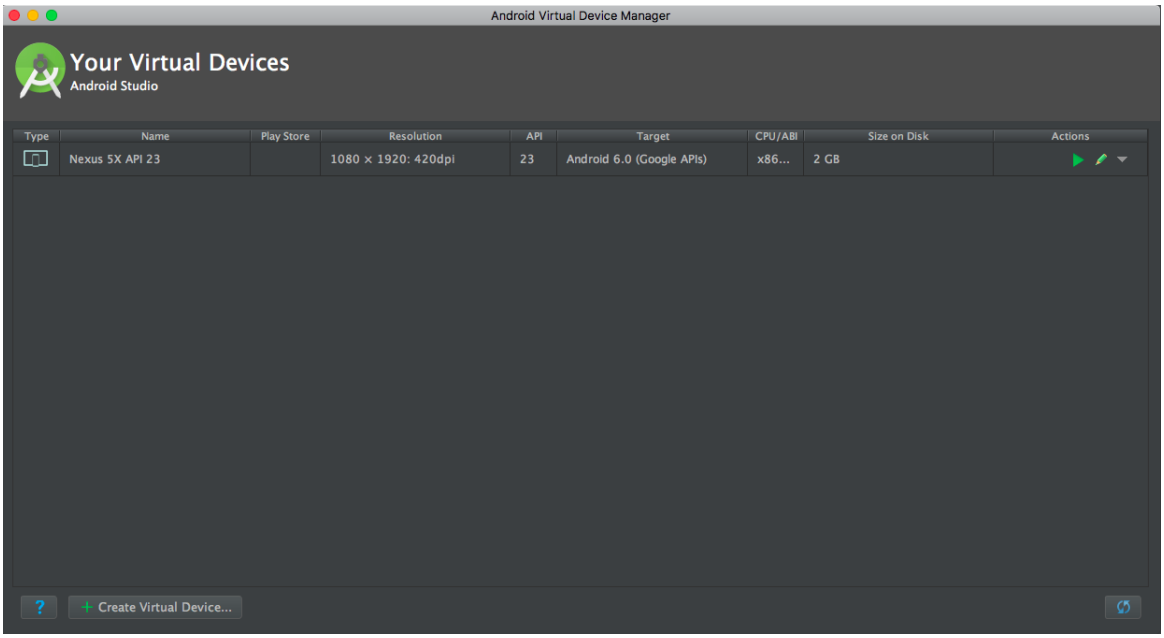



Figura 13 – Gerenciador de Dispositivo Virtual Android

#### 3.2.1.2.1 Android Virtual Device (Emulador Android)

Os dispositivos virtuais criados no Android Studio podem também ser chamados de emuladores. Para iniciar um emulador Android o desenvolvedor possui vários caminhos, mas o principal deles é através do Gerenciador de Dispositivo Virtual Android. A Figura 13 mostra que a ferramenta exibe uma lista com todos os AVDs já criados, e que para cada um deles existe um botão de forma triangular na cor verde, que serve exatamente para iniciar a execução do AVD correspondente.

Na Figura 14, temos o exemplo de um dispositivo virtual Android em execução.


Figura 14 – Emulador de dispositivo móvel com Android

Na lateral do AVD, temos uma barra de ferramentas para executar ações que simulam as de um dispositivo físico Android. Abaixo serão descritas cada uma das ações possíveis através da barra lateral, pela ordem em que os botões nela aparecem.

- Botões para fechar ou minimizar o emulador
- Botão para aumentar o volume.
- Botão para diminuir o volume.
- Botão para girar o posicionamento do AVD no sentido anti-horário
- Botão para girar o posicionamento do AVD no sentido horário
- Botão para capturar uma foto do que está sendo exibido na tela do AVD
- Botão para aumentar zoom no conteúdo exibido na tela do AVD
- Simula o botão físico “Voltar” do AVD.
- Simula o botão físico “Home” do AVD.
- Simula o botão físico “Tarefas” do AVD.
- Abre a tela para simulações avançadas no AVD, onde cada aba permite a inserção de informações para simulação do funcionamento dos diferentes sensores do AVD. Na Fi-

gura 15 a aba “Location” está selecionada, logo, é possível simular diferentes localizações geográficas para o dispositivo.

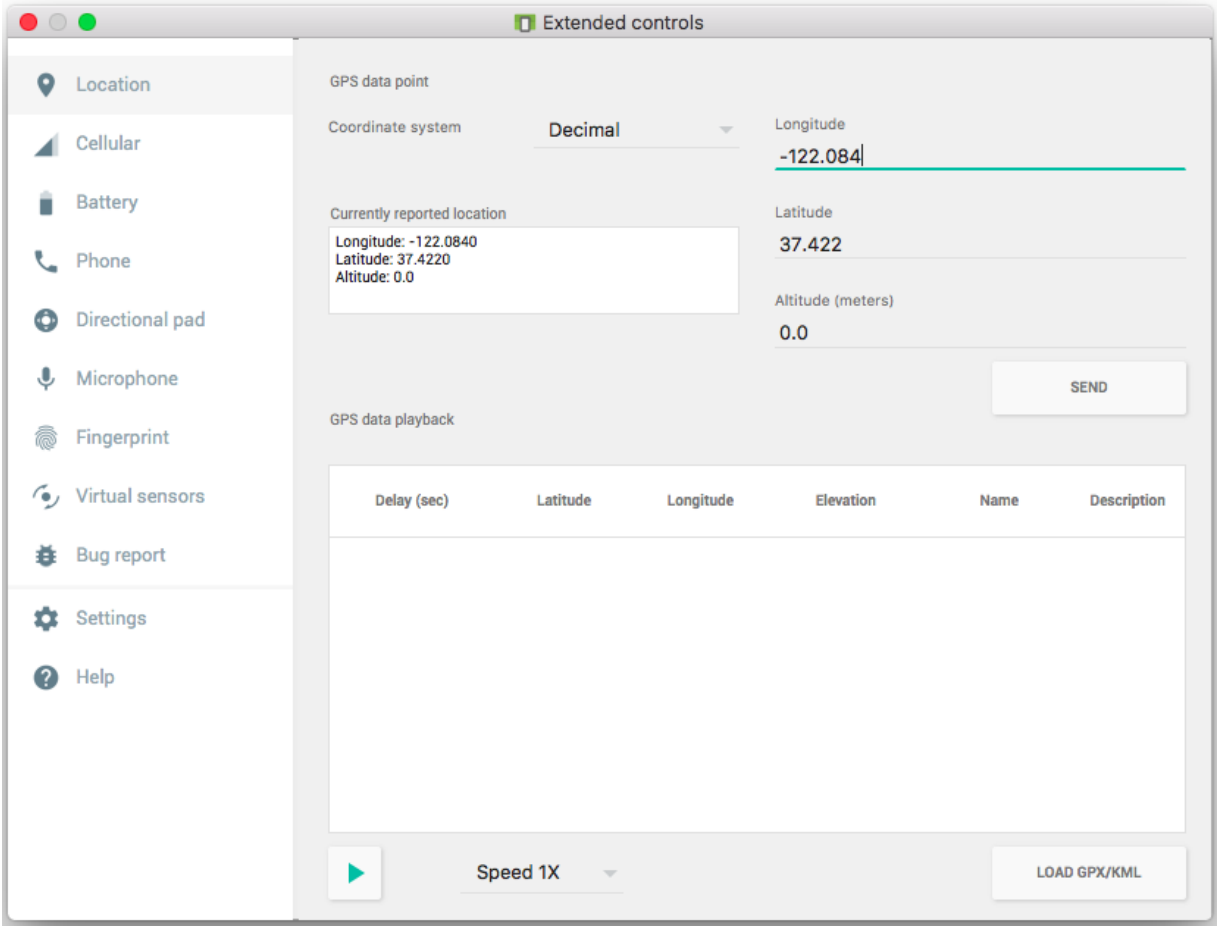



Figura 15 – Configurações de simulação avançadas do AVD

**Criação de um AVD :** Caso não exista nenhum dispositivo virtual disponível na lista do AVD Manager, será necessário que o desenvolvedor faça sua configuração. Abaixo será descrito o processo para criação de um emulador na ferramenta de desenvolvimento oficial do Android.

1. Para a criação de um AVD, o primeiro passo é clicar sobre a opção “Create Virtual Device”, que é encontrada no canto inferior direito na tela do Gerenciador de Dispositivo Virtual Android, o que pode visto na Figura 13.

2. Na sequencia o desenvolvedor será levado para uma tela onde poderá escolher ou personalizar o hardware do dispositivo virtual a ser emulado. Como exibido pela Figura 16, existem opções para se criar dispositivos virtuais que podem emular desde smartphones e tablets ou até mesmo smartwatches e TVs que utilizem o Android como sistema operacional.

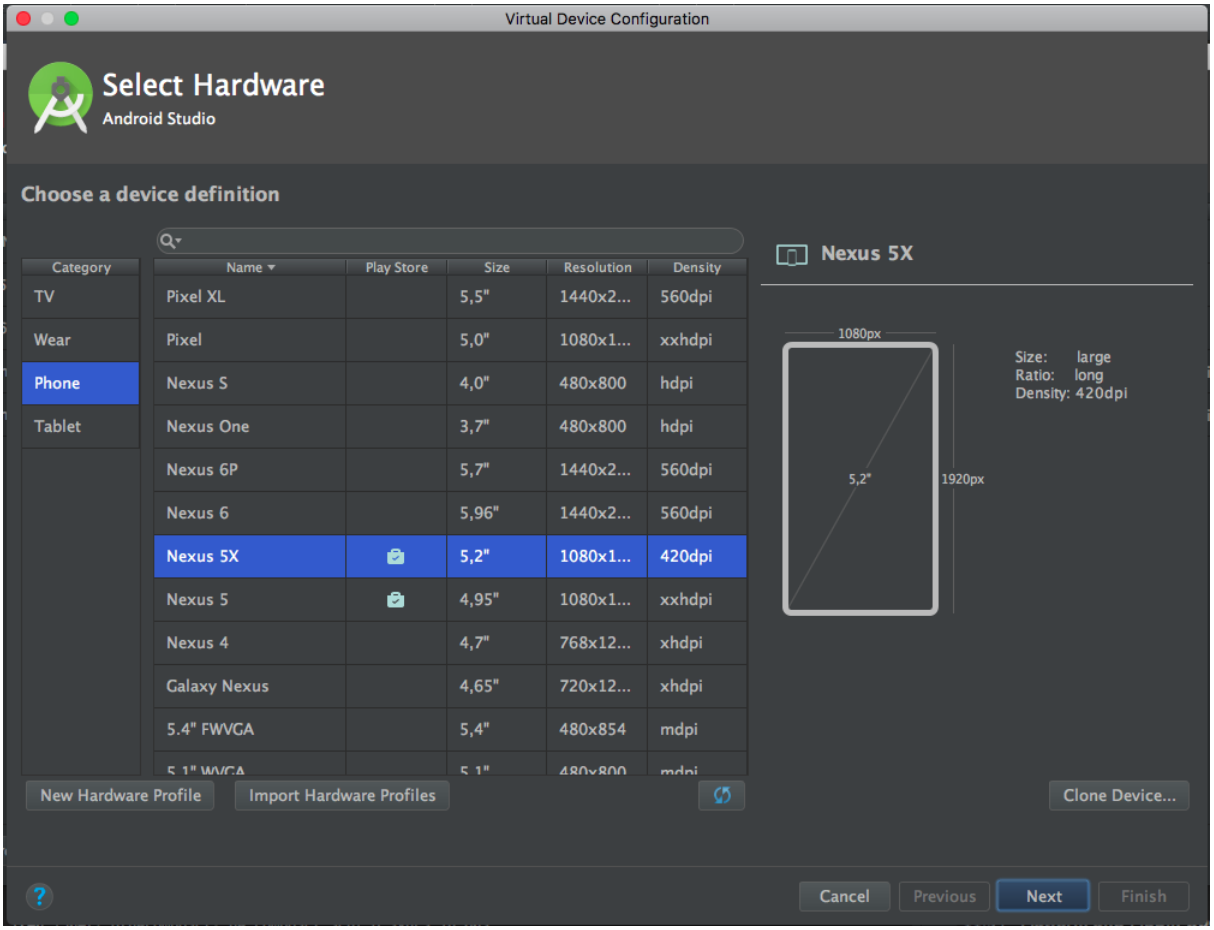



Figura 16 – Criando um Emulador - Escolha do Hardware

3. Quando o desenvolvedor clica sobre o botão “Next” do passo anterior (Figura 16), ele é trazido para a tela de escolha da imagem do Android que será instalada no AVD, veja na Figura 17. O Android Studio disponibiliza imagens de todas as versões existentes do sistema operacional Android, caso a versão escolhida ainda não esteja baixada, basta clicar sobre o link “Download” correspondente à versão desejada e aguardar até que o processo de download da imagem seja concluído. Para finalizar este passo, escolha a imagem que deseja instalar no emulador e clique em “Next”.

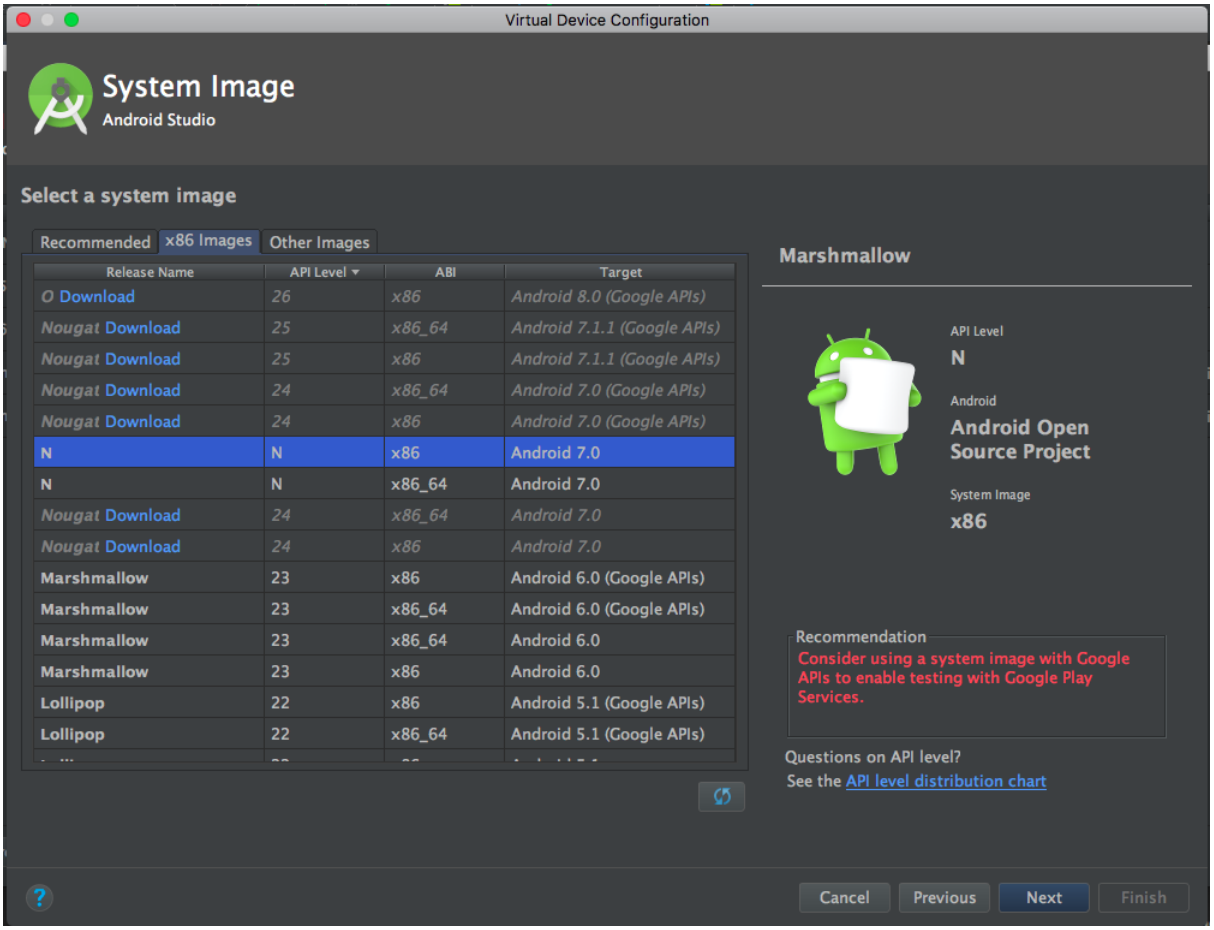



Figura 17 – Criando um Emulador - Escolha da Imagem do Android

4. Por fim, chega-se à tela de configurações do dispositivo virtual (Figura 18), onde é possível dar a ele um nome, escolher a orientação padrão da tela ou realizar outras configurações mais avançadas. Ao clicar sobre o botão “Finish”, as configurações serão aplicadas e o emulador será inserido na lista de dispositivos virtuais disponíveis, veja na Figura 13.

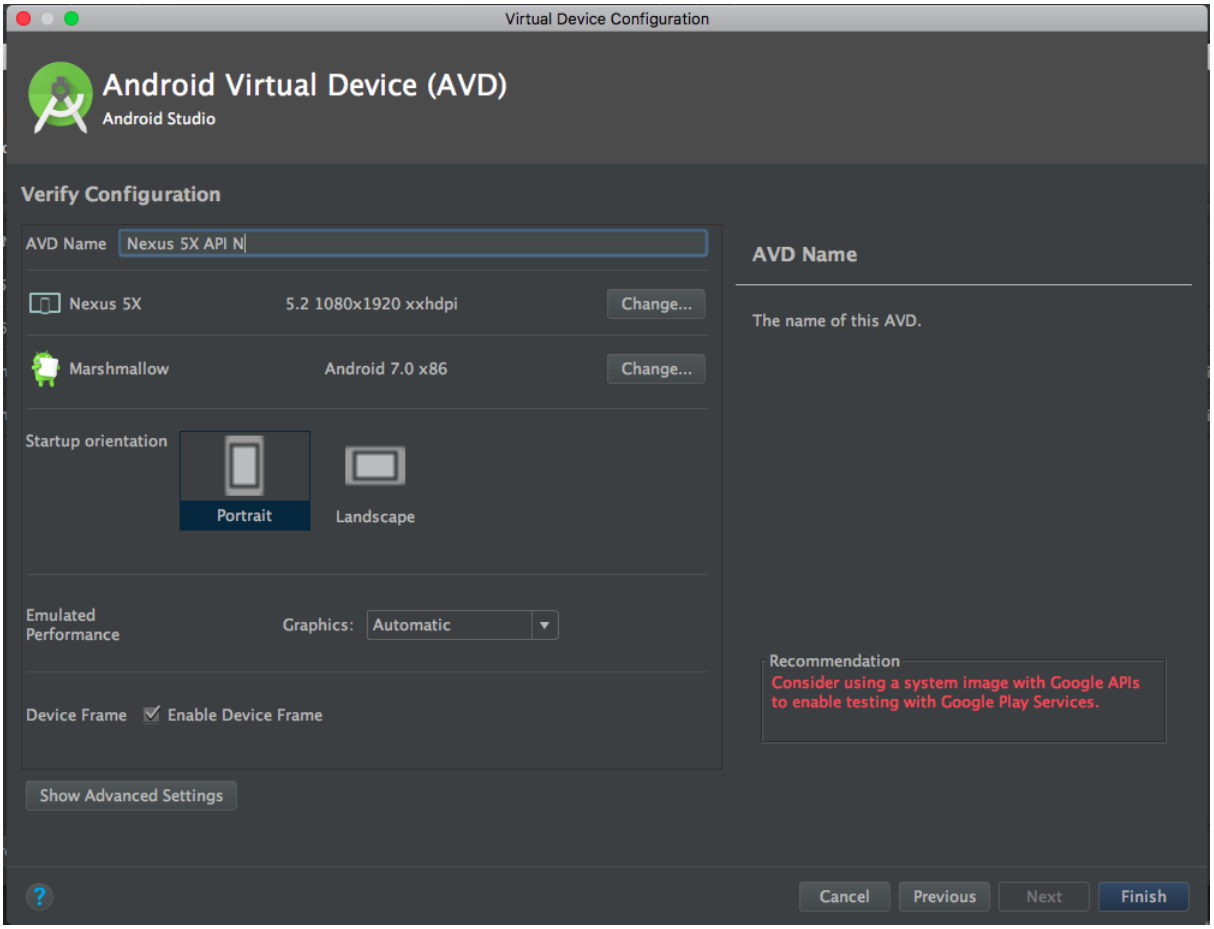



Figura 18 – Criando um Emulador - Configurações Finais

#### 3.2.1.3 Android Device Monitor

A programação é uma tarefa complexa e pode muitas vezes ser desafiadora. Durante o desenvolvimento de um aplicativo o programador se depara com erros de programação que são muito complexos, e que por isso necessitam de ferramentas especializadas que auxiliam encontrar e solucionar esses erros.

No Android Studio, o Android Device Monitor oferece diversas ferramentas de debugging para uso do programador. Com elas, além de encontrar e solucionar os erros de programação, é possível realizar medições do quão custoso computacionalmente é este aplicativo para os dispositivos Android. Em outras palavras, é possível medir entre outros consumos, o uso de memória e de processamento gastos pelo aplicativo. Assim, o programador pode realizar melhorias no código visando aprimorar os custos computacionais do aplicativo, é o que Zapata (2015) aponta em sua obra.

Para utilizar esses recursos o desenvolvedor deve executar o projeto em modo de Debug, o que pode se feito navegando para "Run | Debug 'app'", ou através da barra de ferramentas no botão correspondente. Será exibida uma janela solicitando a escolha o dispositivo virtual ou físico para execução do aplicativo. A partir daí, basta clicar sobre o ícone "Debug" sobre a barra de janela de ferramentas e as janelas serão exibidas para o desenvolvedor, como se vê na Figura 11.

#### 3.2.1.4 Controle de Versão

O controle de versão é uma prática indispensável nos grandes projetos de aplicativos. Quando as versões do código do aplicativo são geridas por uma ferramenta de controle de versão, os desenvolvedores têm mais liberdade para realizar alterações no código do aplicativo pois é possível que todas essas alterações sejam registradas, e caso seja necessário voltar a uma versão anterior do código, o controle de versão facilita o trabalho.

Uma ferramenta completa de controle de versão está atrelada ao Android Studio. Com ela, o desenvolvedor pode realizar o controle de versão do código do projeto por completo, armazenando todas as alterações realizadas no código em uma base local ou também em repositórios na Internet, como por exemplo o GitHub.

#### 3.2.1.5 Gradle

Segundo Berglund e McCullough (2011), o Gradle consiste em uma ferramenta orientada a modelos baseada em Java Virtual Machine, para a construção (build) de um projeto. De forma simplificada, a construção de um projeto dita para o compilador como o módulo do aplicativo (código fonte e arquivos do projeto) e suas dependências devem ser compiladas para criação do arquivo executável final.

No Android, o Gradle tem diversas funcionalidades para auxiliar o desenvolvedor, para utilizá-las é necessário apenas informar em seu arquivo de configuração como se deseja construir o arquivo .apk. Algumas das configurações de build permitem:

- Gerenciamento de Dependências

Controlar as dependências externas que serão utilizadas no aplicativo.

- Compactação de Recursos

  Remove automaticamente todos os recursos, dependências e bibliotecas que estão incluídos no código fonte mas não que não serão utilizados em determinada configuração.

- APKs com Diferentes Funcionalidades ou Configurações

  É possível gerar com o mesmo projeto diferentes versões do aplicativo, por exemplo, uma APK gratuita, com funcionalidades limitadas e outra paga, com todas funcionalidades disponíveis ao usuário. Ou ainda uma versão com resolução para celulares e outra para tablets.

A Figura 19 é um fluxograma que representa o processo de build do Gradle para um projeto típico no Android.

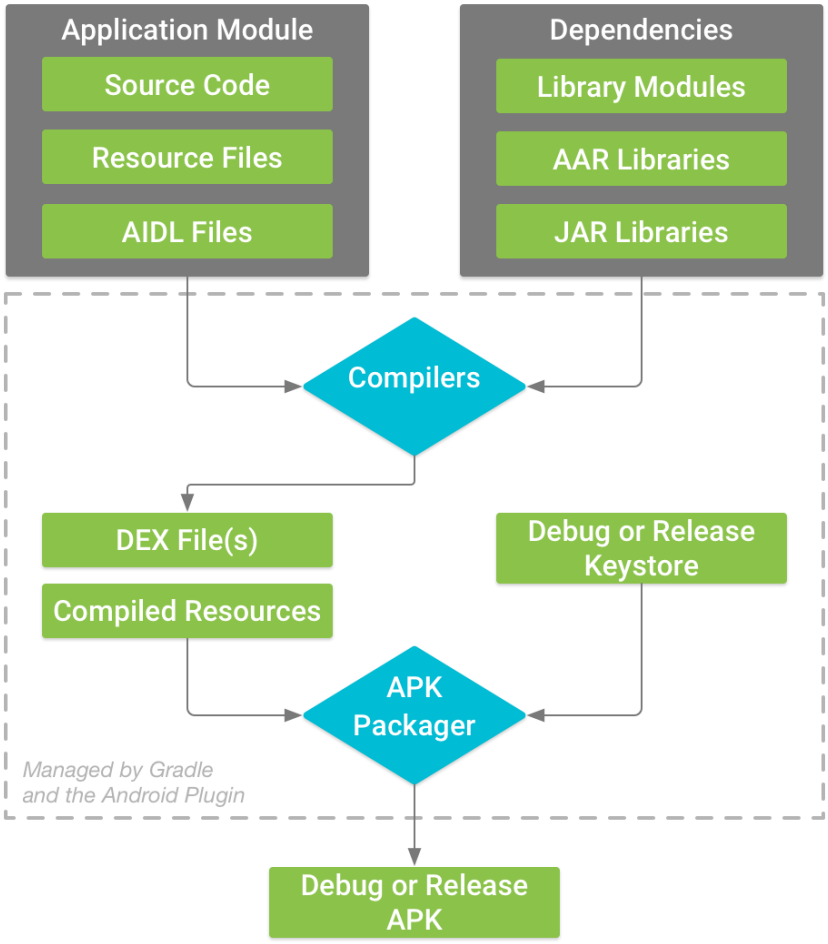



Figura 19 – Processo de construção (build) de aplicativo pelo Gradle (Google Inc., 2016b)

A sequência de passos que descreve o fluxograma da Figura 19 segundo Google Inc. (2016b) é a seguinte:

1. O código fonte é compilado para um arquivo ".dex"(Dalvik Executable).
2. Os arquivos ".dex" e os recursos compilados são combinados pelo APK Packager (Empacotador de APK) em um aquivo APK.
3. O arquivo gerado no passo anterior será agora assinalado pelo APK Packager de acordo com a opção definida pelo desenvolvedor no início do processo de build:
    a) Caso a opção definida tenha sido debug, será incluída uma sinalização para que o APK final possa ser debugado pelo desenvolvedor.
    b) Quando definida a opção de versão de distribuição, o APK será sinalizado para ser distribuído para o usuário final pela Play Store ou outros meios.
4. No ultimo passo o Packeger faz as ultimas modificações no APK por meio de uma ferramenta chamada zipalign para reduzir ao máximo a demanda de memória do aplicativo durante sua execução.

Ao final deste processo de construção com o Gradle, obtêm-se a APK pronta para ser debugada, testada ou liberada para os usuários finais.

### 3.2.2 Fundamentos de Aplicativos Android

Para construir um aplicativo Android, deve-se entender alguns conceitos que são importantes para a criação das funcionalidades necessárias. As questões como declaração das informações de utilização dos recursos do smarthopne e os recursos para criação, desenho e gerenciamento das telas serão os assuntos tratados abaixo.

#### 3.2.2.1 Activity

Quando tocamos sobre o ícone de um aplicativo Android, de maneira genérica, estamos na verdade solicitando ao Sistema Operacional que abra na tela do dispositivo móvel a Activity

principal do app. “Geralmente representa uma tela do app e é responsável por tratar os eventos gerados nesta tela, como, por exemplo, quando o usuário pressiona o botão ou quando um item de menu é escolhido.”,(LECHETA, 2015, p. 93).

#### 3.2.2.1.1 Ciclo de vida da Activity

De acordo com Deitel, Deitel e Wald (2016), uma Activity irá transitar entre um de seus três estados durante o seu ciclo de vida, e isto irá acontecer de acordo com os eventos que ocorrerem. Abaixo serão listados, nas palavras do autor, os estados possíveis de uma Activity:

- Uma Activity *ativa* é visível na tela e tem o foco. Isto é, ela está sendo exibida diretamente na tela e o usuário pode interagir com ela.
- Uma Activity *pausada* é visível na tela mas não tem o foco. Um exemplo é quando um alerta é exibido na tela; neste caso o usuário pode ver a activity no fundo, porém não consegue interagir com ela. isso só será possível quando ele cancelar o alerta, fazendo com que a activity volte ao estado *ativo*.
- Uma activity *parada* não é visível na tela. Ela está em segundo plano e pode ser encerrada pelo sistema caso ele necessite de mais memória. Uma activity se torna *parada* quando outra activity entra em primeiro plano e se torna *ativa*. Um exemplo é quando o usuário recebe uma chamada, o aplicativo de chamadas se torna *ativo* e o app que estava sendo utilizado anteriormente se torna *parado*.

Na Figura 20, temos o fluxograma que representa o ciclo de vida de uma activity em detalhes. Aqui são mostrados os estados da activity após as chamadas dos métodos seus métodos.

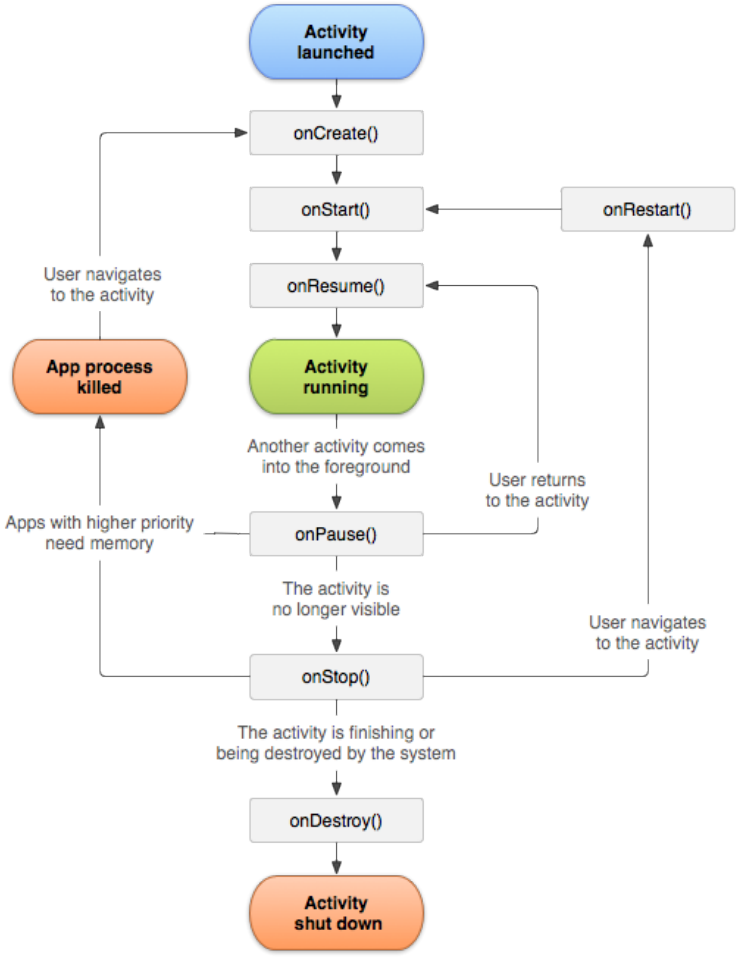



Figura 20 – Ciclo de vida da Activity (Google Inc., 2017b)

Os métodos apresentados na Figura 20 são explicados na Tabela 3, que foi extraída da documentação do Sistema Android.

| Método | Descrição | Pode ser eliminado depois? | Próximo |
|---|---|---|---|
| onCreate() | Chamado quando a atividade é criada pela primeira vez. É onde se deve fazer toda a configuração estática normal — criar exibições, vincular dados a listas etc. Esse método recebe um objeto Bundle (pacote) contendo o estado anterior da atividade, caso esse estado tenha sido capturado (consulte Gravação do estado da atividade mais adiante).<br>Sempre seguido de onStart(). | Não | onStart() |

| Método | Descrição | Pode ser eliminado depois? | Próximo |
|---|---|---|---|
| onRestart() | Chamado depois que atividade tiver sido interrompida, logo antes de ser reiniciada.<br>Sempre seguido de onStart() | Não | onStart() |
| onStart() | Chamado logo antes de a atividade se tornar visível ao usuário.<br>Seguido de onResume() se a atividade for para segundo plano ou onStop() se ficar oculta. | Não | onResume()<br>ou<br>onStop() |
| onResume() | Chamado logo antes de a atividade iniciar a interação com o usuário. Nesse ponto, a atividade estará no topo da pilha de atividades com a entrada do usuário direcionada a ela.<br>Sempre seguido de onPause(). | Não | onPause() |
| onPause() | Chamado quando o sistema está prestes a retomar outra atividade. Esse método normalmente é usado para confirmar alterações não salvas a dados persistentes, animações interrompidas e outras coisas que talvez estejam consumindo CPU e assim por diante. Ele sempre deve fazer tudo bem rapidamente porque a próxima atividade não será retomada até ela retornar.<br>Seguido de onResume() se a atividade retornar para a frente ou de onStop() se ficar invisível ao usuário. | Sim | onResume()<br>ou<br>onStop() |
| onStop() | Chamado quando a atividade não está mais visível ao usuário. Isso pode acontecer porque ela está sendo destruída ou porque outra atividade (uma existente ou uma nova) foi retomada e está cobrindo-a.<br>Seguido de onRestart() se a atividade estiver voltando a interagir com o usuário ou onDestroy() se estiver saindo | Sim | onRestart()<br>ou<br>onDestroy() |
| onDestroy() | Chamado antes de a atividade ser destruída. É a última chamada que a atividade receberá. Pode ser chamado porque a atividade está finalizando (alguém chamou finish() nela) ou porque o sistema está destruindo temporariamente essa instância da atividade para poupar espaço. É possível distinguir entre essas duas situações com o método isFinishing(). | Sim | nada |

| Método | Descrição | Pode ser eliminado depois? | Próximo |
|---|---|---|---|

Tabela 3 – Resumo dos métodos de retorno de chamada do ciclo de vida da atividade (Google Inc., 2017b)

#### 3.2.2.2 AndroidManifest

Para Google Inc. (2017c), o AndroidManifest é, resumidamente, um arquivo que declara ao Sistema Operacional Android os recursos necessários ao aplicativo e suas informações. No caso do arquivo de Manifest do aplicativo desenvolvido neste trabalho, foram declaradas as informações básicas como o nome do aplicativo, o ícone, o tema visual e as Activities utilizadas. Além disso, houve também a declaração das necessidades de permissão para escrever e ler arquivos do armazenamento do dispositivo Android.

Veja parte do código do arquivo AndroidManifest do protótipo desenvolvido:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="me.adolfoquaranta.coletadigital">

    <uses-permission
        android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
        android:name="android.permission.READ_EXTERNAL_STORAGE" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_assignment_turned_in_black_48dp"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity
            android:name=".atividades.Inicio"
            android:label="@string/app_name"
            android:theme="@style/AppTheme.NoActionBar">
        </activity>
        <activity
            android:name=".atividades.CadastroFormulario"
            android:label="@string/title_activity_cadastro_formulario"
            android:theme="@style/AppTheme.NoActionBar" />
        <activity
            android:name=".atividades.CadastroTratamentos"
            android:label="@string/title_activity_cadastro_tratamentos"
            android:theme="@style/AppTheme.NoActionBar" />

        [...]

        <activity
            android:name=".atividades.ListarFormularios"
            android:label="@string/title_activity_listar_formularios"
```

```
            android:parentActivityName=".atividades.Inicio"
            android:theme="@style/AppTheme.NoActionBar">
            <meta-data
                android:name="android.support.PARENT_ACTIVITY"
                android:value="me.adolfoquaranta.coletadigital.atividades.Inicio" />
        </activity>
        <activity
            android:name=".atividades.ListarColetas"
            android:label="@string/title_activity_listar_coletas"
            android:theme="@style/AppTheme.NoActionBar" />

        <activity
            android:name=".atividades.SplashScreenActivity"
            android:theme="@style/AppCompat.TelaCheia">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

#### 3.2.2.3 Intent

Os componentes executáveis do Android necessitam de uma forma de se comunicar e trocarem informações. Com essa finalidade, o SO, os serviços de segundo plano e as activities utilizam-se de um mecanismo denominado Intent. Deitel, Deitel e Wald (2016) explica que quando o programador tem a "intenção" que o Android execute alguma tarefa pré-programada, basta que ele diga isto ao Android por meio de uma Intent e o sistema irá cuidar de encontrar o caminho para realizar a tarefa.

##### 3.2.2.3.1 Comunicação entre Apps

A ação de compartilhar uma fotografia e um exemplo prático da aplicação das Intents para comunicação entre Apps. Quando o usuário escolhe a opção compartilhar no aplicativo, existe uma Intent definida pelo programador dentro do código do Aplicativo que será executada pelo Sistema Operacional. O Android irá "entender" por meio desta Intent, que o usuário tem a intenção de compartilhar uma foto e por isso irá exibir para ele uma tela com todos os aplicativos que estão disponíveis para compartilhamento de imagens.

#### 3.2.2.3.2 Executar Aplicativos ou Iniciar Activities

Outra funcionalidade das Intents é auxiliar o programador a executar um aplicativo ou uma nova Activity a partir de uma ação do usuário. Para isso, primeiro o desenvolvedor deve criar uma Intent especificando qual é o aplicativou ou a Activity que ele deseja iniciar. Depois, basta utilizar o método startActivity(), que irá receber a Intent previamente definida e interpretar as informações lá contidas para executar a ação desejada pelo usuário.

#### 3.2.2.3.3 Determinar qual Activity Executar

As Intents também trabalham em conjunto com o arquivo AndroidManifest, apresentado em subseção 3.2.2.2. Para exemplificar está função da Intent, a ultima Activity declarada naquele trecho de código será transcrita abaixo:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="me.adolfoquaranta.coletadigital">

    [...]

        <activity
            android:name=".atividades.SplashScreenActivity"
            android:theme="@style/AppCompat.TelaCheia">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    [...]

</manifest>
```

Repare que existe uma declaração entre as linhas <*intent-filter*> e </*intent-filter*>. Esta declaração é denominada “Filtro de Intent” e utiliza as Intents para mostrar ao Sistema Operacional Android que a Activity “SplashScreenActivity” será a primeira a ser executada quando o utilizador do aplicativo abri-lo.

### 3.2.2.4 View

A palavra da língua inglesa, View, pode ser traduzida para o português como a palavra ver, ou visão. Isto indica muito sobre sua função no Android. “A classe **android.view.View** é a classe-mãe de todos os componentes visuais do Android, e suas diversas subclasses são utilizadas para criar a interface gráfica das telas.” (LECHETA, 2015, p. 162).

Sendo assim, no cotexto do usuário, todos os botões, entradas e visualizadores de texto, entre outros componentes das telas de um aplicativo, são uma view. Já no contexto do desenvolvedor, View é a classe programada em Java que trabalha com os componentes que o usuário irá visualizar na tela.

### 3.2.3 SQLite

O SQLite é uma biblioteca que implementa uma engine de base de dados, ferramenta utilizada para trabalhar com a persistência dos dados das aplicações da plataforma Android, neste caso. Em outras palavras, o SQLite trabalha para armazenar os dados do aplicativo, de forma forma estruturada e organizada, facilitando o acesso e manipulação aos dados por parte do desenvolvedor.

Segundo Lecheta (2015), o Android disponibiliza integrado ao seu SDK, uma API para utilização do SQLite, sem a necessidade de softwares ou ferramentas externas. Apesar disso, as demais opções também não são descartadas, o método a ser utilizado vai depender da preferência do desenvolvedor.

### 3.2.4 Linguagem Java

O Java é uma linguagem de programação de computadores desenvolvida pela Sun Microsystems em 1991. O objetivo principal do Java, quando foi criado, era possibilitar que o desenvolvedor escrevesse um programa em Java e que ele fosse compatível com a maioria dos microcomputadores existentes naquela época.

O objetivo da Sun foi alcançado, pois, apesar do Java ser uma linguagem de programação orientada a objetos que o código necessita ser compilado, o resultado da compilação é um arquivo que será interpretado pela Java Virtual Machine, um dos principais componentes da plataforma Java, que possibilita essa portabilidade dos programas.

Atualmente o Java vem sendo fortemente utilizado no setor corporativo, principalmente em sistemas e servidores web. Além disso, Deitel e Deitel (2017) aborda que o Java é a principal linguagem utilizada atualmente no desenvolvimento dos aplicativos móveis para a plataforma

Android.

### 3.2.5 Bibliotecas externas MaterialEditText, MaterialSpinner e opencsv

O Android SDK permite ao desenvolvedor utilizar bibliotecas externas que irão complementar o pacote de desenvolvimento de um app Android. As bibliotecas externas são um conjunto de códigos, no Android programadas em Java, criados para facilitar a vida dos desenvolvedores. As bibliotecas normalmente fornecem soluções para problemas recorrentes, diminuindo o retrabalho do programador.

Neste aplicativo, foram utilizadas as bibliotecas MaterialEditText, MaterialSpinner e opencsv, que facilitam o trabalho com as entradas de texto, com as caixas de seleção de texto flutuantes e com a geração de arquivos csv, respectivamente. Para que as bibliotecas pudessem ser utilizadas, elas foram adicionadas como dependências do projeto, como foi explicado na subseção 3.2.1.5.

Veja as importações das bibliotecas para o aplicativo desenvolvido na Figura 21, que começam onde existe uma marcação com o simbolo de uma lampada:

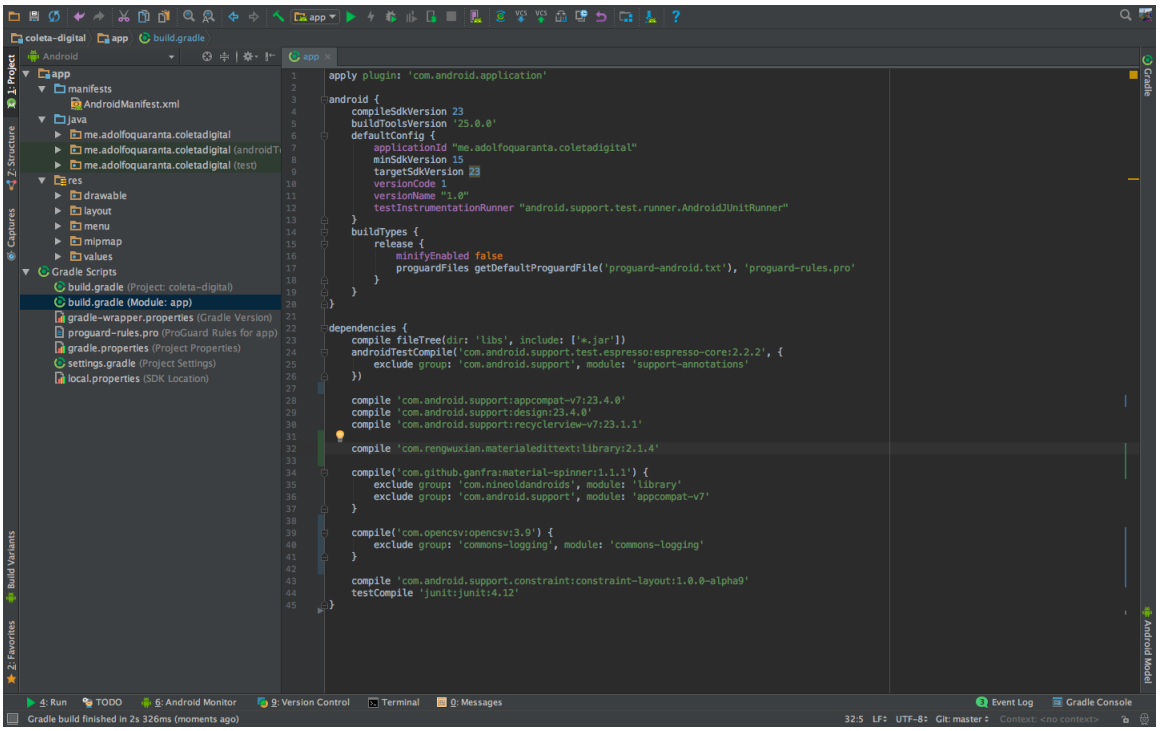



Figura 21 – Importação da Bibliotecas Externas (Arquivo de configuração do Gradle)

# 4 Resultados

Nas primeira seção do capítulo de resultados, serão apresentados os diagramas criados na fase de projeto do protótipo do aplicativo Coleta Digital. Além da primeira, outra seção será dedicada a um exemplo de uso do app, criado para mostrar parte das funcionalidades e das telas do protótipo.

## 4.1 Modelagem do Software

A modelagem do software através de diagramas é uma parte fundamental do processo de criação de softwares. Os diagramas auxiliam o desenvolvedor e o cliente no entendimento das necessidades e funcionalidades do projeto. Nesta seção serão apresentados os diagramas criados para este projeto.

### 4.1.1 Diagrama de Casos de Uso

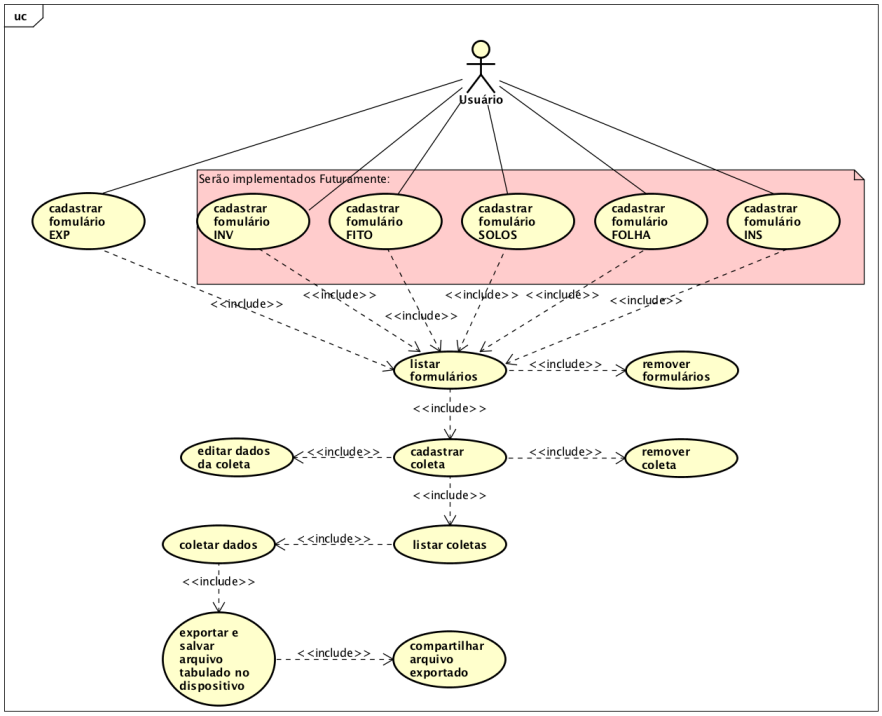



Figura 22 – Diagrama de Casos de Uso

### 4.1.2 Diagrama de Classes

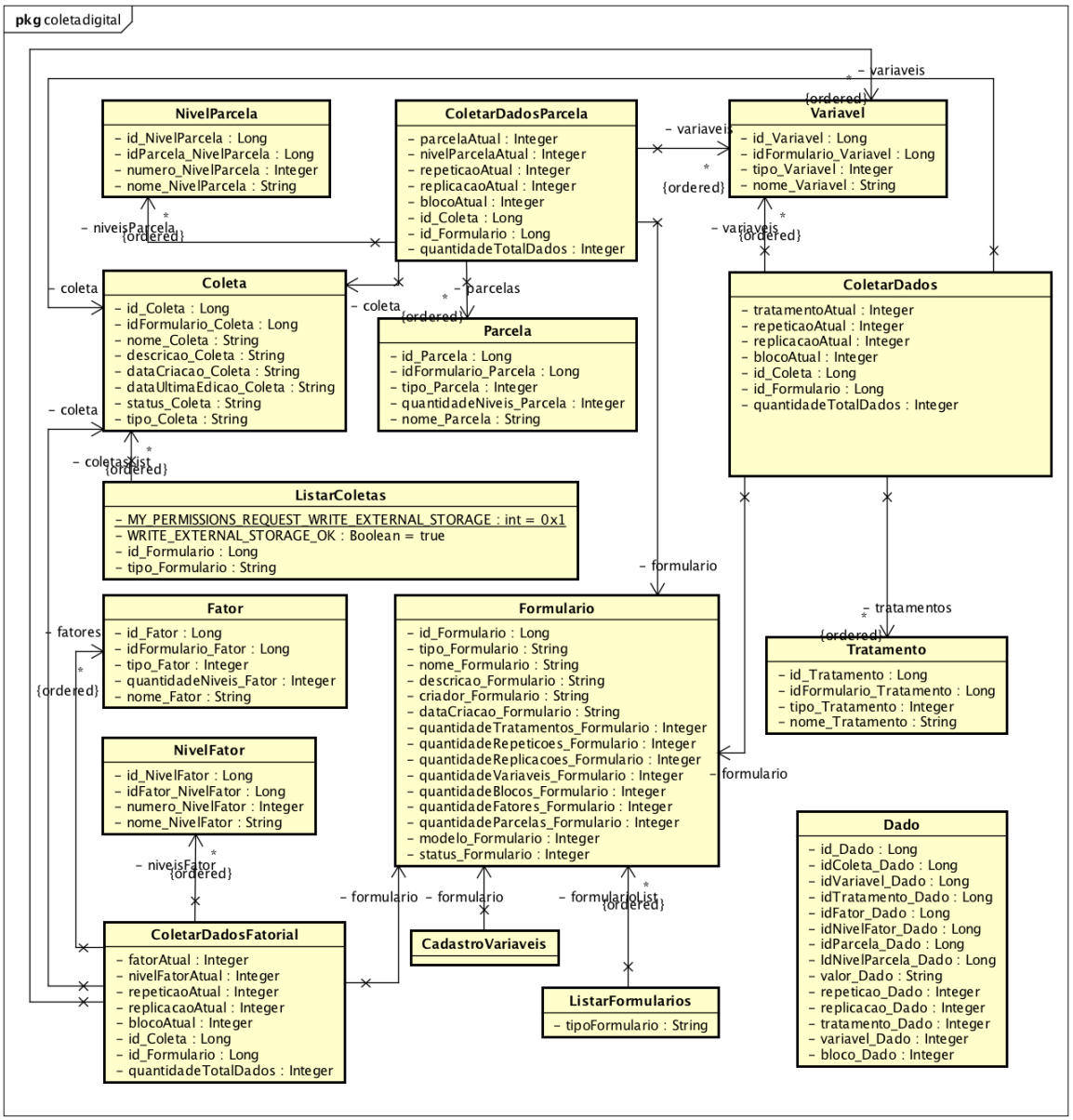



Figura 23 – Diagrama de Classes

### 4.1.3 Modelo Entidade Relacionamento

A base de dados é um dos componentes mais importantes para este aplicativo. É nela queque toda informação coletada armazenada de forma estruturada e concisa. Isso possibilitará ao utilizador do aplicativo trabalhar com os dados armazenados pelo tempo que for necessário, por exemplo, quando ele necessitar comparar dois resultados de coletas realizadas com um grande espaço de tempo ele elas.

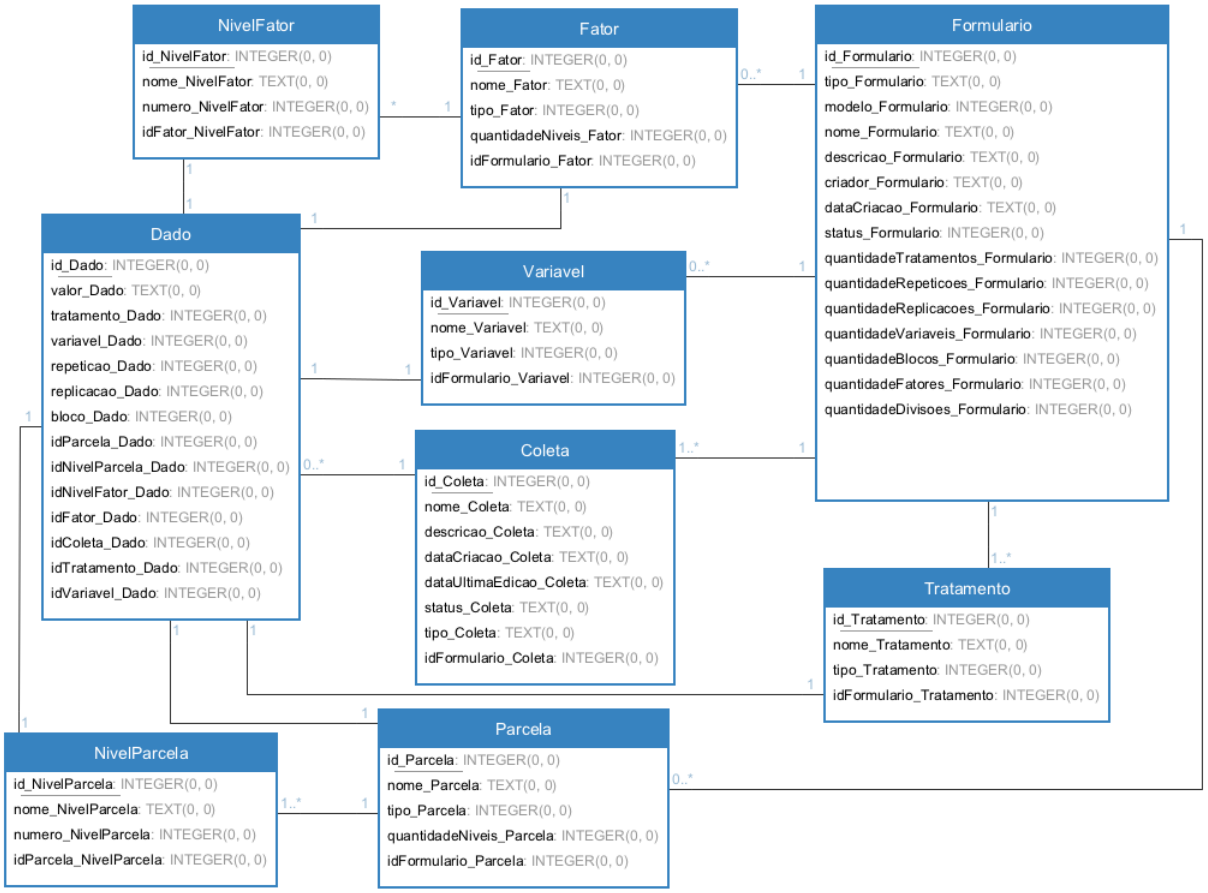



Figura 24 – Modelo Entidade Relacionamento

## 4.2 O protótipo de aplicativo

O desenvolvimento de um aplicativo é uma tarefa complexa e requer, na maioria das vezes, uma equipe com um número considerável de desenvolvedores para ser concluída. Como este trabalho foi realizado individualmente, resolveu-se desenvolver o protótipo de um aplicativo, o que significa que o projeto apresentado é um esboço do aplicativo final. Sendo assim, apenas

algumas das funcionalidades foram completamente desenvolvidas, as demais serão criadas em trabalhos futuros.

### 4.2.1 Telas Iniciais

Por questões de aparência, convencionou-se o uso de uma tela de apresentação do aplicativo que é exibida por alguns segundos antes da tela inicial do aplicativo. O nome para este tipo de tela é um termo do inglês, splashscreen, e pode ser vista na Figura 25.

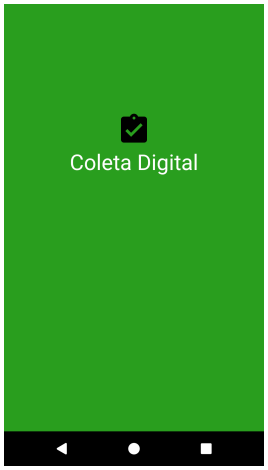



Figura 25 – Tela de apresentação

Depois de 2,5 segundos de exibição da tela de apresentação do aplicativo, o usuário será direcionado para a tela inicial, como apresentado na Figura 26. Nesta tela o utilizador terá uma série de botões, onde cada um deles levará para o cadastro ou a listagem dos formulários correspondentes ao experimento representado pelo botão.

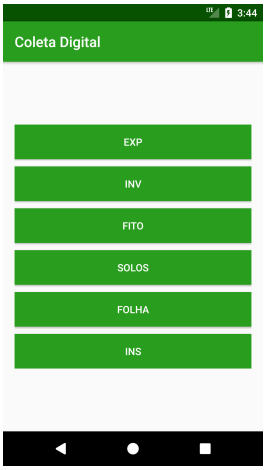



Figura 26 – Tela inicial

Quando o usuário seleciona o botão “EXP” na tela inicial (Figura 26), existem dois caminhos possíveis. O primeiro caminho ocorre caso não haja nenhum outro experimento cadastrado, então o utilizador será levado para a tela de cadastro de um novo “EXP” (Figura 27a).

O segundo caminho ocorre caso já exista experimentos cadastrados, então o usuário será levado para a tela que lista estes experimentos (Figura 30).

### 4.2.2 Cadastro de "EXP" (Exemplo com "DIC")

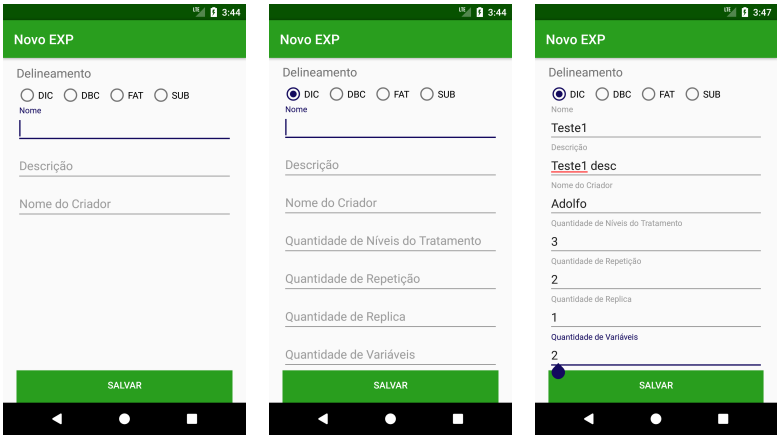
Para que o usuário do aplicativo possa coletar os dados do "EXP" é necessário que ele faça primeiro o cadastro das informações do arranjo utilizado em seu ensaio.

O exemplo do cadastro que será descrito abaixo servirá apenas para demonstrar o funcionamento do aplicativo e consistirá em um Delineamento Inteiramente Casualizado com 3 níveis para o Tratamento, 2 Repetições, 1 Réplica e 2 Variáveis. Veja a descrição passo a passo:

#### 4.2.2.1 Cadastro das informações iniciais do "EXP" com "DIC"

A Figura 27a mostra a primeira tela exibida ao usuário para o cadastro de um novo formulário de coleta. Os campos exibidos na tela irão mudar de acordo com a escolha do delineamento feita pelo usuário, que neste exemplo, como já mencionado será um "DIC". Repare que quando o item "DIC" foi selecionado (Figura 27b), novos campos foram inseridos na tela.

A tela com todas as informações básicas do experimento preenchidas é mostrada pela Figura 27c. Conforme as entradas vão sendo preenchidas, o aplicativo faz a validação dos dados inseridos de acordo com as especificações de cada uma delas. Será exibido um erro abaixo da entrada caso ela seja preenchida com algum dado inválido ou não seja preenchida.




(a) (b) (c)

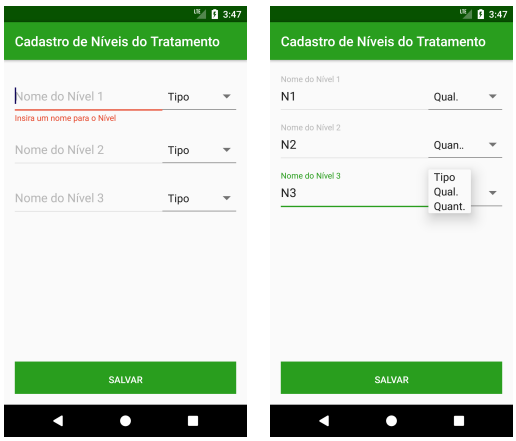
Figura 27 – Cadastro das informações iniciais do "EXP" com "DIC"

Ao final do preenchimento, o usuário deverá clicar sobre o botão “Salvar”, assim o app irá verificar todos os campos por algum dado invalido e caso esteja tudo correto o usuário será redirecionado para a tela de cadastro dos níveis do tratamento (Figura 28), caso contrário serão exibidos avisos dos erros que necessitam ser corrigidos.

#### 4.2.2.2 Cadastro dos Níveis do Tratamento

A montagem da tela do cadastro dos níveis de tratamento depende da informação de “Quantidade de Níveis do Tratamento”, que foi inserida na etapa anterior. A Figura 28a exibe a montagem da tela baseada no exemplo proposto no inicio desta seção, onde existem 3 Níveis para o Tratamento.

Na referida tela existem 3 linhas, cada uma dessas linhas representa um dos níveis do tratamento, e para cada um desses níveis o usuário deverá inserir um nome e um “Tipo” para aquele nível, como foi feito na Figura 28b. A informação “Tipo” que deverá ser inserida para cada um dos níveis nesta etapa está relacionada à natureza do nível a ser avaliado estatisticamente, o que poderá ser qualitativo ou quantitativo.




(a) (b)

Figura 28 – Cadastro dos níveis do Tratamento

Todos os níveis devem possuir um nome e um tipo, e essas informações serão validadas em dois momentos distintos. A primeira validação ocorre individualmente para cada campo de texto ou seleção existente na tela, logo após seu preenchimento. A segunda validação é realizada após o toque em “Salvar”, e verifica o conjunto de todas as informações existente na tela, caso

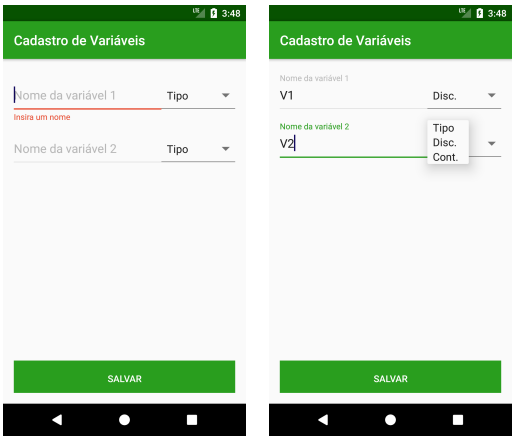
tudo esteja válido o usuário poderá prosseguir para o cadastro das variáveis, caso contrário antes de prosseguir deverá corrigir os erros apontados.

#### 4.2.2.3 Cadastro das Variáveis

O cadastro de Variáveis que vemos na Figura 29a é muito semelhante ao cadastro dos níveis de tratamento pois a montagem dessa tela pelo aplicativo também depende de uma informação provida no primeiro passo do cadastro de um "EXP", que neste caso é a "Quantidade de Variáveis".

Além da semelhança anterior, cada uma das variáveis, assim como os níveis do tratamento, exigem um nome e um "Tipo", com a diferença que o tipo da variável poderá ser discreta ou contínua, o que irá variar de acordo com a natureza estatística do dado produzido no experimento que será avaliado.

De acordo com o exemplo criado para demostrar este cadastro, sabe-se que o experimento irá analisar 2 Variáveis, logo teremos duas linhas para inserção dos nomes e dos tipos das variáveis a serem analisadas.




(a) (b)

Figura 29 – Cadastro das Variáveis

Assim como nas etapas anteriores, podemos ver na Figura 29b a tela com todas as informações preenchidas e validadas, logo o utilizador pode tocar sobre o botão "Salvar" e finalizar o cadastro do seu ensaio, o que irá leva-lo para a tela com a lista de experimentos cadastrados (Figura 30).

### 4.2.3 Lista de Experimentos

A função desta tela é mostrar ao usuário do aplicativo uma lista com todos os experimentos já cadastrados na base de dados do aplicativo. Além disso, existe um botão azul com o sinal de adição no canto inferior direito da tela, como mostra a Figura 30, o que permite adicionar novos experimentos nesta lista.

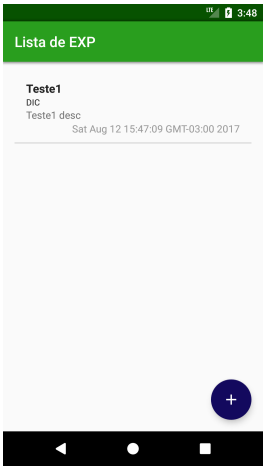



Figura 30 – Lista de experimentos cadastrados

Quando o utilizador tocar sobre um dos itens da lista, ou seja um dos experimentos cadastrados, ele poderá visualizar uma tela com 3 opções para aquele item selecionado, veja a Figura 31.

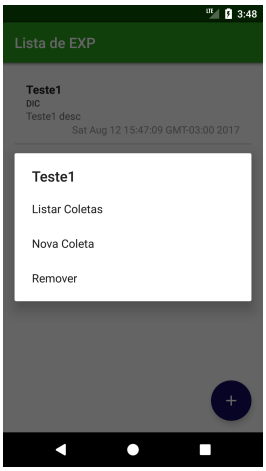



Figura 31 – Ações possíveis a partir de um experimento da lista

As opções “Nova Coleta” e “Listar Coletas”, por serem mais complexas serão mostradas na subseção 4.2.5 e na subseção 4.2.5, respectivamente. Já a opção “Remover”, é mais simples e será mostrada a seguir.

#### 4.2.3.1 Remover

A opção remover, por ser mais simples exibe apenas mais uma tela, onde informa ao utilizador que a ação de remoção de um experimento não poderá ser desfeita e pergunta se realmente é o que ele deseja fazer. Para que a resposta possa ser dada, existem ainda dois botões, "Cancelar" e "Remover", que executam exatamente as ações de seus nomes, veja na Figura 32.

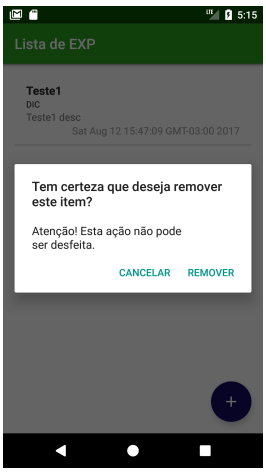
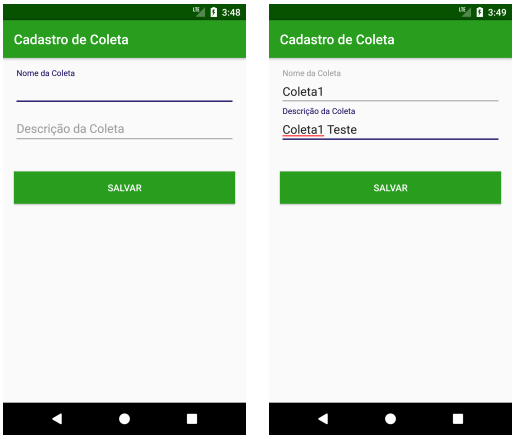



Figura 32 – Tela de confirmação de remoção de um experimento

### 4.2.4 Cadastro Coleta

Durante o decorrer de um experimento, o pesquisador terá a necessidade de realizar diversas medições dos dados ali produzidos. Portanto o aplicativo deve possibilitar que essas diferentes mensurações para o mesmo experimento possam ser registradas, criando assim um histórico das medições realizadas. Para isso, criou-se dentro do aplicativo um conceito de coleta que servirá para identificar cada uma dessas medições realizadas.

Sendo assim, antes de poder realizar a coleta dos dados é necessário primeiro fazer o cadastro das informações da coleta, como na Figura 33, pois estas informações serão importantes para a identificação das coletas. E importante ressaltar que o cadastro de uma coleta só é possível depois de realizado o cadastro de um "EXP", pois as coletas são sempre atreladas a um experimento.




(a) (b)

Figura 33 – Cadastro de uma coleta

Ao finalizar o preenchimento das informações para o cadastro de uma coleta, o utilizador do aplicativo será redirecionado para a tela de coleta dos dados, como mostra a Figura 38a.

### 4.2.5 Lista de Coletas

A tela com a lista de coletas, como o próprio nome indica, irá mostrar para o utilizador as coletas cadastradas para um determinado "EXP". O usuário do aplicativo poderá ser trazido para esta tela através da escolha da opção "Listar Coletas" (Figura 31).


Figura 34 – Lista de coletas

Nesta tela, se o usuário tocar em qualquer uma das coletas presente na lista, ele terá uma tela idêntica a que está representada na Figura 35, onde apenas o título irá variar de acordo com a coleta selecionada. As opções disponíveis para qualquer uma das coletas serão

"Exportar", "Editar/Continuar" ou "Remover". A opção "Exportar" será abordada em detalhes na subseção 4.2.7, já as duas ultimas opções, por serem menos densas, serão explicadas abaixo.

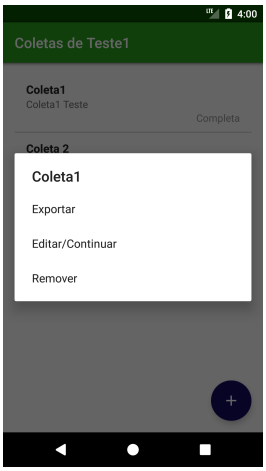



Figura 35 – Ações sobre um item da lista de coletas

#### 4.2.5.1 Remover

Para remover uma coleta o usuário do aplicativo deverá ser seguir um processo que é semelhante ao de remoção de um "EXP". Primeiro ele deve selecionar qual das coletas deseja remover da lista de coletas (Figura 34) e consequentemente da base de dados. Depois a tela com as opções de ações sobre a coleta (Figura 35) será exibida e ele deve selecionar a opção remover. Ao final, será exibida uma tela com uma mensagem e os botões de confirmação ou cancelamento da remoção da coleta, como se vê na Figura 36.

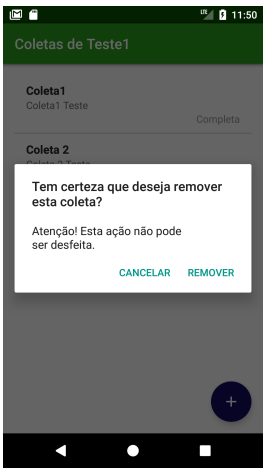



Figura 36 – Confirmação de remoção de um item da lista de coletas

#### 4.2.5.2 Editar/Continuar

A opção “Editar/Continuar” foi criada para duas ocasiões diferentes que podem ocorrer com uma coleta. Porém as duas utilizam o mesmo princípio, que será enviar o utilizador do aplicativo para a tela de coleta de dados (Figura 37).

A primeira será útil quando o utilizador perceber que em uma coleta que já foi completada existe um erro nos dados salvos, então ele deverá “Editar” aquela coleta já completada.

A segunda deverá ser utilizada no caso de uma coleta de dados que já foi iniciada e por algum motivo o usuário resolveu interromper a coleta antes do final, neste caso, o utilizador escolheu a opção “Continuar” e então será enviado para o ponto da coleta em que parou.

### 4.2.6 Coletar Dados (Exemplo de “EXP” com “DIC”)

A funcionalidade coletar dados é uma das mais importantes no aplicativo. A tela representada na Figura 37 guiará o usuário de forma coordenada e intuitiva, fazendo com que ele insira todos os dados necessários nos campos determinados.

Veja abaixo os componentes da tela:

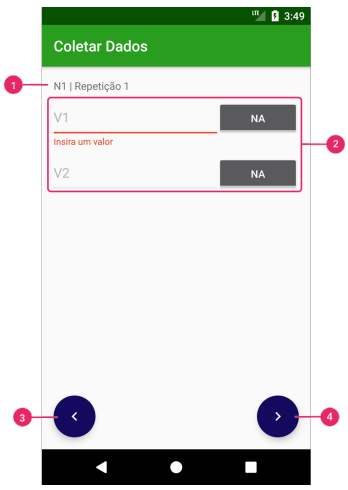



Figura 37 – Tela de coleta de dados

1. Texto que informa em qual posição da coleta o usuário está.

2. Painel de entrada de valores para as variáveis analisadas.

3. Botão para retroceder na coleta

4. Botão para avançar na coleta

O funcionamento da tela de coleta é baseado nas ações tomadas pelo utilizador e nas quantidades de Níveis do Tratamento, Repetições, Réplicas e Variáveis do “EXP” selecionado para coleta. Logo, para explicar esta funcionalidade, o procedimento para uma coleta do “EXP” cadastrado na subseção 4.2.2 foi recriado na Figura 38.

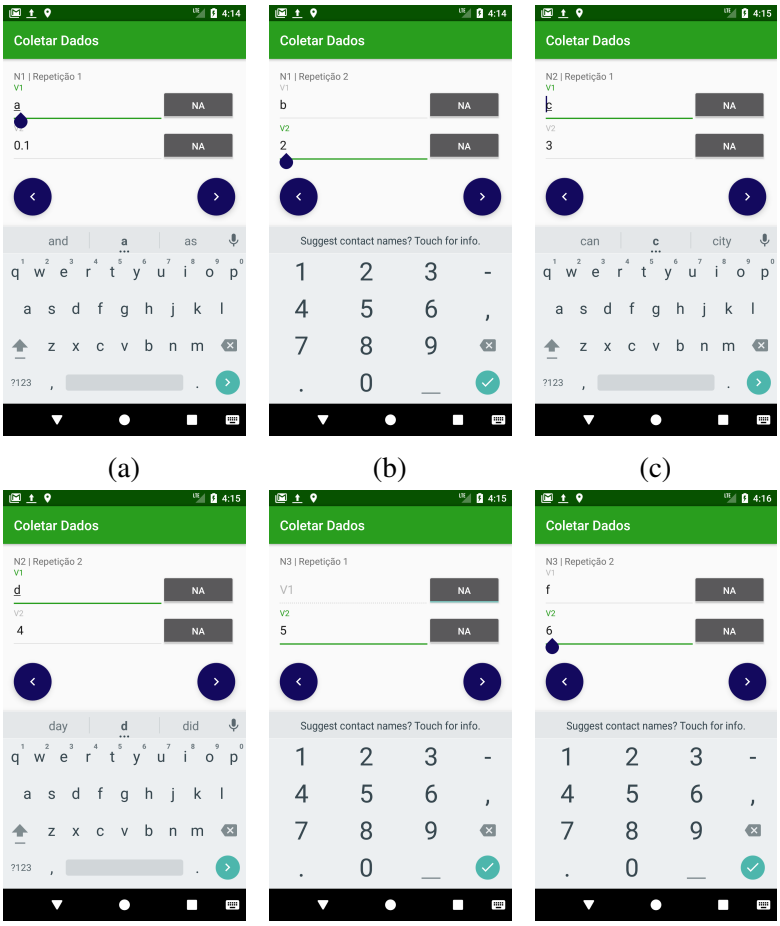



Figura 38 – Exemplo dos passos de uma coleta de dados

A dinâmica simplificada de coleta do exemplo apresentado segue o seguinte algoritmo quando o usuário seleciona o botão avançar na coleta:

(a) Nível do Tratamento=N1, Repetição = 1, Réplica = 1

- salva os dados inseridos para as variáveis

- incrementa Repetição

(b) Nível do Tratamento=N1, Repetição = 2, Réplica = 1

- salva os dados inseridos para as variáveis
- Repetição = 1
- incrementa Nível do Tratamento

(c) Nível do Tratamento=N2, Repetição = 1, Réplica = 1

- salva os dados inseridos para as variáveis
- Incrementa Repetição

(d) Nível do Tratamento=N2, Repetição = 2, Réplica = 1

- salva os dados inseridos para as variáveis
- Repetição = 1
- incrementa Nível do Tratamento

(e) Nível do Tratamento=N3, Repetição = 1, Réplica = 1

- salva os dados inseridos para as variáveis
- Repetição = 1

(f) Nível do Tratamento=N3, Repetição = 2, Réplica = 1

- salva os dados inseridos para as variáveis
- final da coleta

Note que cada elemento da combinação entre Nível do Tratamento, Repetição e Replica representa um item a ser estudado no experimento, e que para cada item, são coletados os valores para todas as variáveis definidas no cadastro do "EXP".

Caso o usuário utilize o botão de retroceder na tela de coleta, a aplicativo ira decrementar os valores que foram incrementados no algoritmo anterior, fazendo o caminho inverso na coleta. Os valores das variáveis que já haviam sido coletados serão reexibidos, e caso o usuário necessite, ele poderá alterá-los.

Repare neste caso, o teclado exibido ao selecionar o campo para inserção do valor da Variável "V1" (Figura 38a) é diferente do teclado exibido quando o campo de inserção do valor

da Variável “V2” (Figura 38b) é escolhido. O teclado exibido será condizente com o tipo de dado da variável, que foi definido pelo usuário quando ele criou o “EXP”.

Outro detalhe que deve ser destacado é o botão “NA”, que tem a função de marcar como desconsiderado o valor que seria inserido no campo de texto a sua esquerda. Na Figura 38e, o botão “NA” foi ativado, logo o campo de texto da Variável V1, que é o campo à sua esquerda, foi desativado e consequentemente desconsiderado. Esta função é importante no caso da morte de uma muda, por exemplo.

### 4.2.7 Exportar

Tudo que foi feito até o momento serve de base para a criação do arquivo tabulado que servirá para alimentar um software de analise estatística. Todos os dados produzidos pelo ensaio que o pesquisador deseja avaliar, agora serão formatados e exportados em um arquivo. Este arquivo poderá ser salvo na memória local do smartphone, salvo em serviços de nuvem ou compartilhado com outros pesquisadores.

Para exportar o arquivo, o utilizador deverá selecionar a opção “Exportar”, que é disponibilizada para qualquer item exibido na lista de coletas, mostrada pela Figura 35.

#### 4.2.7.1 Permissão e Salvamento Local

As versões mais recentes do Sistema Operacional Android gerenciam as permissões de utilização dos recursos do dispositivo de uma forma mais amigável e intuitiva. As permissões são solicitadas apenas no momento em que elas se fazem necessárias. Este recurso utilizado no aplicativo, fazendo com que a permissão para acesso e gravação na memória do dispositivo seja solicitado na primeira vez que usuário escolher exportar os dados de uma coleta, como mostra a Figura 39.

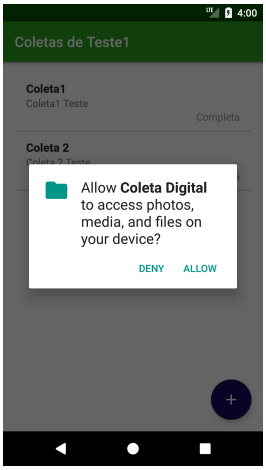
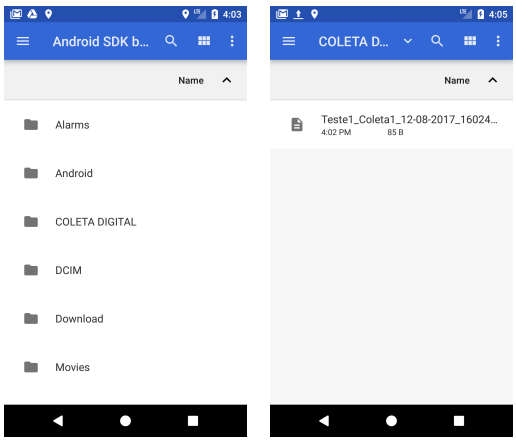



Figura 39 – Permissão para gravar no armazenamento do smartphone

Depois que a permissão de gravação na memória interna do dispositivo for concedida, o usuário poderá acessar o arquivo com o uso do gerenciador de arquivos disponível no dispositivo Android. O primeiro passo, que pode ser visto na Figura 40a, é encontrar a pasta "COLETA DIGITAL" que estará localizada no diretório raiz do dispositivo Android.

Ao selecionar a pasta "COLETA DIGITAL", o usuário tera todas as coletas que já foram exportadas, nomeadas por uma concatenação do nome do "EXP", com o nome da coleta e com a data da exportação. Veja na Figura 40b.




(a) (b)

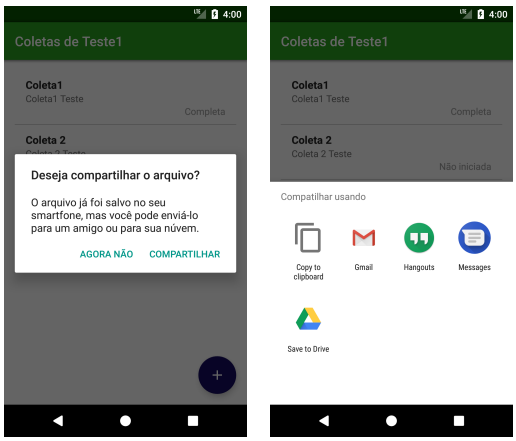
Figura 40 – Encontrando o arquivo salvo no armazenamento local no gerenciador de arquivos

#### 4.2.7.2 Compartilhamento do Arquivo

De maneira muito simples, o app permitirá que o compartilhamento do arquivo gerado seja feito com outras pessoas ou com serviços de armazenamento na nuvem. Logo depois de

exportar os dados, o aplicativo irá questionar o utilizador se ele deseja compartilhar o arquivo que foi exportado, como exibido na tela representada na Figura 41a.

Caso a opção "Compartilhar" seja escolhida, o usuário sera redirecionada para a tela que disponibilizará todas as possibilidades de compartilhamento do arquivo gerado, veja na Figura 41b. Caso a opção escolhida seja "Agora não", o arquivo será salvo somente na memória local do dispositivo.



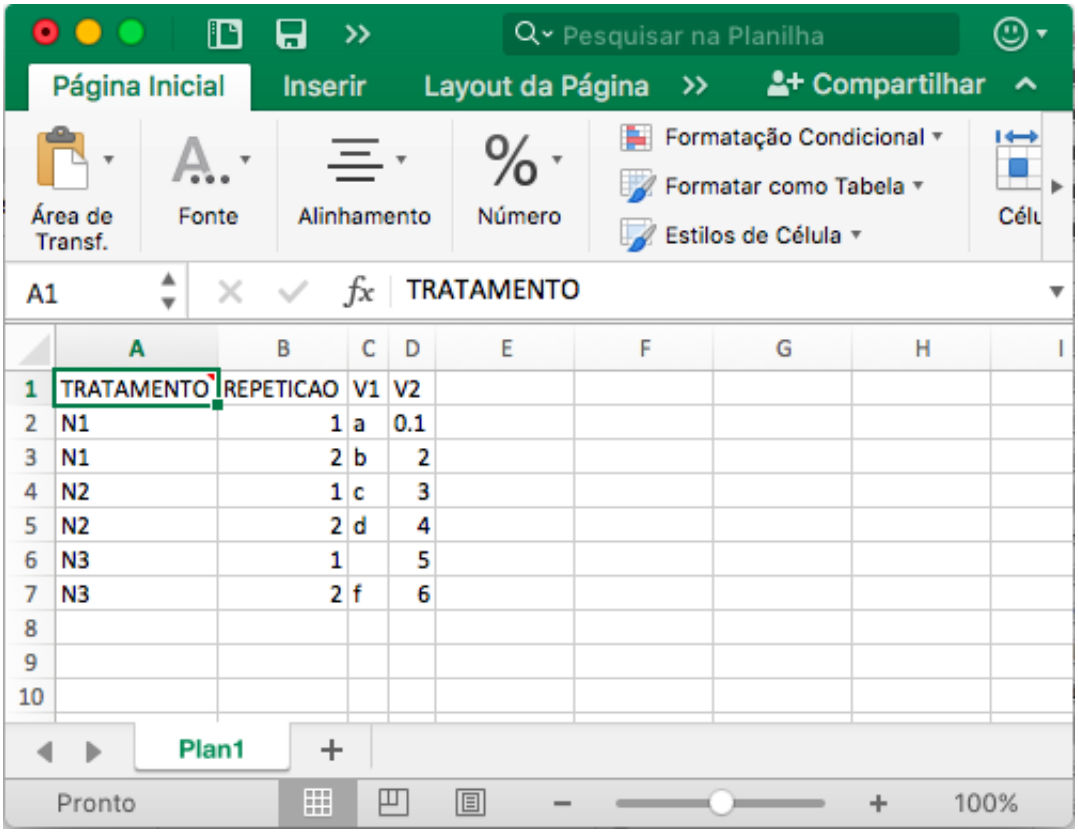

(a) (b)

Figura 41 – Exemplo dos passos para compartilhar o arquivo gerado

#### 4.2.7.3 O arquivo gerado (Exemplo de "EXP" com "DIC")

O resultado final do arquivo gerado pelo aplicativo será como está disposto na Figura 42. A tabulação dos dados coletados é realizada seguindo os seguintes quesitos:

- Cada linha da tabela representa uma muda
  - A primeira linha será o cabeçalho do arquivo
- A primeira coluna terá os Níveis do Tratamento
- A segunda coluna terá as repetições para cada Nível
- A terceira coluna terá as réplicas para cada Nível
  - Como existe apenas 1 réplica no "EXP", a coluna foi omitida.
- As demais colunas terão os valores das variáveis




| | A | B | C | D |
|---|---|---|---|---|
| 1 | TRATAMENTO | REPETICAO | V1 | V2 |
| 2 | N1 | 1 | a | 0.1 |
| 3 | N1 | 2 | b | 2 |
| 4 | N2 | 1 | c | 3 |
| 5 | N2 | 2 | d | 4 |
| 6 | N3 | 1 | | 5 |
| 7 | N3 | 2 | f | 6 |
| 8 | | | | |
| 9 | | | | |
| 10 | | | | |

Figura 42 – O arquivo gerado para o exemplo proposto

## 4.3 Repositório do Protótipo

O controle das versões do código deste protótipo foi realizado e está disponível através do link https://github.com/adolfoquaranta/coleta-digital.

# Considerações Finais

Partindo do problema inicial proposto, que era o método ultrapassado de coleta de dados dos experimentos estatísticos de campo da Engenharia Florestal, foram realizados estudos sobre a aplicabilidade de tecnologias móveis para a solução de problemas nas Ciências Agrárias. Com base em estudos de diversos trabalhos semelhantes, notou-se que os resultados obtidos através da aplicabilidade da tecnologia móvel são sempre muito positivos, o que demonstrou e justificou a necessidade e viabilidade da construção desse aplicativo.

Durante o projeto e desenvolvimento do protótipo, foram enfrentadas grandes dificuldades, como problemas na formulação da base de dados, dos algorítimos de coleta dos dados, falhas na estrutura da tabulação dos dados. Contudo, todas as adversidades foram contornadas e ao final do trabalho obteve-se um resultado satisfatório.

Como apresentado, foi possível utilizar a Plataforma Android na construção de uma solução para o problema proposto. O protótipo de aplicativo é funcional e pode atender às necessidades da Engenharia Florestal para coleta, manipulação e armazenamento dos dados, além da exportação e compartilhamento de um arquivo com os dados tabulados.

Finalmente, para trabalhos futuros, podem ser sugeridas algumas melhorias necessárias para o funcionamento completo do protótipo, veja os itens abaixo:

- Melhorias na estrutura da base de dados.
- Melhorias na estrutura visual e de design das telas.
- Melhorias na funcionalidade de tabulação e exportação dos dados.
- O desenvolvimento da funcionalidade de cadastro dos demais tipos de ensaios, como o mostrado no diagrama da subseção 4.1.1.

# Bibliografia

BAMBINI, M.; LUCHIARI-JÚNIOR, A.; ROMANI, L. Mercado de Aplicativos Móveis (apps) para uso na Agricultura. In: *SIMPÓSIO NACIONAL DE INSTRUMENTAÇÃO AGROPECUÁRIA*. São Carlos, SP: EMBRAPA, 2014. v. 1, n. 1, p. 711–714. Disponível em: <http://ainfo.cnptia.embrapa.br/digital/bitstream/item/112339/1/mercado.pdf>. Citado na página 15.

BERGLUND, T.; MCCULLOUGH, M. *Building testing with Gradle*. First edit. Sebastopol, US: O'Reilly Media, Inc., 2011. v. 53. 110 p. ISSN 1098-6596. ISBN 9781449304638. Citado na página 43.

COSTA, N. P. O.; FILHO, N. F. D. Análise E Avaliação Funcional De Sistemas Operacionais Móveis: Vantagens E Desvantagens. *Revista de Sistemas e Computação*, Salvador, v. 3, n.1, p. 66–77, 2013. Disponível em: <http://www.revistas.unifacs.br/index.php/rsc/article/download/2581/1950>. Citado na página 27.

DEITEL, P.; DEITEL, H. *Java: como programar*. 10 ed.. ed. São Paulo: Pearson Education do Brasil, 2017. 934 p. ISBN 9788543019055. Citado na página 52.

DEITEL, P.; DEITEL, H.; WALD, A. *Android 6 for Programmers - An App-Driven Approach*. third ed. Boston: Prentice Hall, 2016. 422 p. ISBN 9780134289366. Citado 3 vezes nas páginas 33, 46 e 50.

DESCARDECI, R. Comunicações por Satélite – Técnicas de Transmissão, Multiplexação e de Acesso. v. 04, p. 1–17, 2001. Disponível em: <http://www.inatel.br/revista/busca/45-comunicacoes-por-satelite-tecnicas-de-transmissao-multiplexacao-e-de-acesso-s893783-1/file>. Citado na página 23.

G1. *Operadora lança G1, primeiro celular com a plataforma Google*. 2008. Disponível em: <http://g1.globo.com/Noticias/Tecnologia/0,,MUL770366-6174,00-OPERADORA+LANCA+G+PRIMEIRO+CELULAR+COM+A+PLATAFORMA+GOOGLE.html>. Citado 2 vezes nas páginas 7 e 21.

Google Inc. *Android 6.0 Marshmallow*. 2015. Disponível em: <https://developer.android.com/about/versions/marshmallow/index.html>. Citado na página 32.

Google Inc. *Conheça o Android Studio*. 2015. Disponível em: <https://developer.android.com/studio/intro/index.html?hl=pt-br>. Citado 3 vezes nas páginas 7, 34 e 36.

Google Inc. *Android 7.0 Nougat*. 2016. Disponível em: <https://developer.android.com/about/versions/nougat/index.html>. Citado na página 32.

Google Inc. *Configure sua compilação*. 2016. Disponível em: <https://developer.android.com/studio/build/index.html?hl=pt-br>. Citado 3 vezes nas páginas 7, 44 e 45.

Google Inc. *The Android Source Code*. 2016. Disponível em: <https://source.android.com/source/index.html>. Citado 2 vezes nas páginas 7 e 28.

Google Inc. *Android O Developer Preview*. 2017. Disponível em: <https://developer.android.com/preview/index.html>. Citado na página 33.

Google Inc. *Atividades*. 2017. Disponível em: <https://developer.android.com/guide/components/activities.html?hl=pt-br>. Citado 4 vezes nas páginas 7, 9, 47 e 49.

Google Inc. *Manifesto do aplicativo*. 2017. Disponível em: <https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=pt-br>. Citado na página 49.

JAIN, V. S. et al. Overview on Generations of Network : 1G , 2G , 3G , 4G , 5G. v. 5, n. 5, p. 1789–1794, 2014. Disponível em: <http://www.ijcta.com/documents/volumes/vol5issue5/ijcta2014050534.pdf>. Citado 2 vezes nas páginas 22 e 24.

JUNIOR, J. C. V.; VENTURA, L. A. G. Automação do processo de pulverização de defensivos e mapeamento de plantio da produção. *Journal of Chemical Information and Modeling*, Curitiba - PR, v. 53, n. 9, p. 1689–1699, nov 2011. ISSN 1098-6596. Disponível em: <http://www.up.edu.br/blogs/engenharia-da-computacao/wp-content/uploads/sites/6/2015/06/2011.14.pdf>. Citado na página 16.

LECHETA, R. R. *Google Android - Aprenda a criar aplicações para dispositivos móveis com o Android SDK*. 2 ed.. ed. São Paulo: Novatec, 2015. 608 p. ISBN 9788575224687. Citado 3 vezes nas páginas 46, 51 e 52.

OLIVEIRA, R. S. D.; CARISSIMI, S.; TOSCANI, S. Sistemas Operacionais. *Revista de Informática Teórica e Aplicada*, VIII, n. 3, p. 1–33, 2001. ISSN 0103-4308. Citado na página 27.

OTSUKA, G. S.; ZANELATO, A. P. A. O sistema Android no universo dos dispositivos móveis. *Encontro De Iniciação Científica - ETIC*, Vol. 8, n. No 8, 2012. Disponível em: <http://intertemas.toledoprudente.edu.br/revista/index.php/ETIC/article/view/3759/3520>. Citado 2 vezes nas páginas 15 e 21.

PAGOTO, R. *Comunicação D2D Em Redes 5g: Desafios*. 2016. Disponível em: <http://www.decom.ufop.br/imobilis/comunicacao-d2d-em-redes-5g-desafios/>. Citado 5 vezes nas páginas 7, 23, 24, 25 e 26.

PAULA, L. J. L. de. *Desenvolvimento de aplicativo para dispositivos móveis para coleta de dados georreferenciados através de reconhecimento de voz*. Tese (Doutorado) — Universidade de São Paulo - Escola Superior de Agricultura Luiz de Queiroz, Piracicaba - SP, jul 2013. Disponível em: <http://www.teses.usp.br/teses/disponiveis/11/11152/tde-10062013-091453/>. Citado na página 16.

PRETTO, S. J. Coleta de dados com dispositivos móveis: Um estudo de caso aplicado à produção avícola. Lageado - RS, jun 2013. Disponível em: <http://hdl.handle.net/10737/384>. Citado na página 16.

SANTOS, G. H. dos; SPIRLANDELLI, L. P.; GOTARDO, R. A. AddressDroid: Apresentação do Desenvolvimento e Funcionamento de uma Aplicação Orientada a Objetos para Android. *Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica*, v. 2, n. n. 1, 2012. Disponível em: <http://periodicos.unifacef.com.br/index.php/resiget/article/view/291>. Citado na página 15.

SHARMA, P. Evolution of Mobile Wireless Communication Networks-1G to 5G as well as Future Prospective of Next Generation Communication Network. *International Journal of Computer Science and Mobile Computing (IJCSMC) - not index*, v. 2, n. 8, p. 47–53, 2013.

Disponível em: <http://www.ijcsmc.com/docs/papers/August2013/V2I8201317.pdf>. Citado na página 24.

SIMÃO, A. M. d. L. Proposta de método para análise pericial em smartphone com sistema operacional android. 2011. Disponível em: <http://repositorio.unb.br/handle/10482/9938>. Citado 2 vezes nas páginas 20 e 21.

SOOD, R.; GARG, A. Digital Society from 1G to 5G : A Comparative Study. v. 3, n. 2, p. 186–193, 2014. Disponível em: <http://ijaiem.org/volume3issue2/IJAIEM-2014-02-27-065.pdf>. Citado na página 25.

Sue Shellenbarger. *The '80s Called, and They Want Their Cellphones Back*. 2012. Disponível em: <https://www.wsj.com/articles/SB10001424052702303812904577297763321318488>. Citado 2 vezes nas páginas 7 e 20.

TANENBAUM, A. S.; WOODHULL, A. S. *Sistemas Operacionais: Projetjos e Implementação*. 3. ed. Porto Alegre: Bookman, 2008. 993 p. Citado na página 27.

TSENG, F. M. et al. An integrated model for analyzing the development of the 4G telecommunications market in Taiwan. *Telecommunications Policy*, v. 38, n. 1, p. 14–31, 2014. ISSN 03085961. Citado 2 vezes nas páginas 7 e 22.

VEJA. *Apple celebra dez anos do iPhone, o celular que mudou a telefonia*. 2017. Disponível em: <http://veja.abril.com.br/economia/apple-celebra-dez-anos-do-iphone-o-celular-que-mudou-a-telefonia/>. Citado 2 vezes nas páginas 7 e 21.

WAKU, G. M. et al. A Robust Software Product Line Architecture for Data Collection in Android Platform. In: UNIVERSITY OF CAMPINAS (UNICAMP). *IX Brazilian Symposium on Components, Architectures and Reuse Software*. Belo Horizonte - MG: Institute of Electrical & Electronics Engineers IEEE, 2015. p. p. 31 – 39. Disponível em: <http://dx.doi.org/10.1109/SBCARS.2015.14>. Citado na página 15.

ZAPATA, B. C. *Android Studio Essentials*. Birmingham, UK: Packt Publishing, 2015. 126 p. ISSN 1098-6596. ISBN 978-1-78439-720-3. Citado na página 42.